FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Parallel computing on RNA-Sequencing processes

Daniel João Lopes Moreira

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Jorge Manuel Gomes Barbosa

Co-supervisor: Rui Camacho

July 18, 2016

Parallel computing on RNA-Sequencing processes

Daniel João Lopes Moreira

Mestrado Integrado em Engenharia Informática e Computação

July 18, 2016

Abstract

Medicine is an area which have been always under a constant evolution with an increasing number of researches being performed every day. With the permanent evolution new medical models came to light, such as the personalized medicine, which advocates that medical care should be individually optimized for each patient. However, this would only be possible by knowing every person's unique variation of the human genome. For this knowledge to be gathered the development of efficient and cheap human genome sequencing techniques was required. Two of those techniques include Microarrays and RNA-Sequencing analysis.

Despite providing the best results, as far as studying gene expression and detecting differential expression genes is concerned, the RNA-Sequencing method's usage is being limited by its costs, which are higher than the ones associated with Microarrays, becoming impractical for large studies. This way the current dissertation aims at improving its performance in order to reduce its *makespan* and consecutively its execution costs.

Due to the fact that the tools which execute the RNA-Seq analysis are already parallelized, taking advantage of multicore architectures, the idea of distributing the computation among multiple computers came to light, coupled with the implementation of a scheduler in order to, provide ideally the best schedule possible, so that the developed system is able of executing every RNA-Sequencing task in the most convenient node of the cluster, distributing the computation needed to be done.

The implemented solution, revealed very efficient at improving the performance of the RNA-Sequencing tasks, by reducing its executions times. In the performed tests that aimed at validating the developed system, using only two nodes of a cluster to execute this processes, every tool's execution time have been considerably reduced, getting up to a 50% reduction in the *HTSeq* one.

Resumo

A medicina é uma área que tem, ao longo dos tempos, estado sujeita a uma constante evolução, sempre com um número de pesquisas e investigações crescente. A permanente evolução levou a que novos modelos médicos fossem criados, como por exemplo a medicina personalizada que defende que os tratamentos médicos devem ser individualmente personalizados para cada um dos doentes. No entanto tal modelo só poderia ser implementado se houvesse o conhecimento sobre as características únicas de cada paciente. Para este efeito havia a necessidade de criar técnicas eficientes capazes da obter tais informações. Duas destas técnicas criadas são os *Microarrays* e a sequenciação do genoma através de moléculas RNA.

Apesar da segunda alternativa oferecer melhores resultados, no que diz respeito ao estudo da expressão dos genes bem como à deteção de genes de expressão diferenciada, a utilização do método de sequenciação do genoma tem sido limitada devido aos elevados custos agregados à sua execução, que são consideràvelmente maiores que os custos associados aos Microarrays, tornando impraticável a sua utilização em grandes experiências. Desta forma o principal objectivo desta dissertação é o de melhorar o desempenho dos processos de sequenciação de genomas, reduzindo o seu tempo de execução e automaticamente os seus custos.

Derivado do facto de que as ferramentas usadas para executar cada etapa da sequenciação do genoma já se encontrarem paralelizadas, tirando partido da arquitectura multicore, a ideia de distribuir a computação por um conjunto de computadores, surgiu acopolada à implementação de um escalonador com o objectivo de fornecer o escalonamento ideal das tarefas, para que posteriormente o sistema desenvolvido fosse capaz de as executar no nó mais conveniente do cluster, distribuindo assim a computação associada à sequenciação de genomas.

A solução implementada, revelou-se bastante eficaz no que diz respeito ao aumento de desempenho deste tipo de tarefas, através da redução significativa dos seus tempos de execução. Nos testes realizados com o objectivo de validar o sistema desenvolvido, mesmo usando apenas dois nós do cluster para executar as tarefas de sequenciação de genomas, para todas as ferramentas testadas uma redução considerável do tempo de execução foi obtida, chegando até aos 50% de redução para a ferramenta *HTSeq*.

Acknowledgements

I owe many thanks for completing this dissertation.

First, I want to thank my supervisor Jorge Manuel Gomes Barbosa, for all the supports, guidance and ideas provided along this whole process, as well as the valuable time spent in meetings.

My gratitude to Rui Camacho Ferreira da Silva for all the feedback and for giving me all the necessary conditions to successfully validate my thesis as well as for the help provided regarding the cluster's configuration.

I would like to thank to some of my closest friends including Jennifer Moleiro, Hugo Sousa, Marie Ros, Vasco Gomes, Hugo Cardoso and Francisco Maciel, for all the kindness and for always supporting me, helping me to overcome the hard times. A big thank to them as well, for all the amazing relaxing moments shared and the working sessions at FEUP.

Last, but not least, a big thanks to my family for always supporting me and my choices both emotionally and financially.

Daniel Moreira

"You'll always miss 100% of the shots you never take."

Wayne Gretzky

Contents

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

CPU	Central Processing Unit
DAG	Directed acyclic graph
DNA	Deoxyribonucleic acid
GPU	Graphics Processing Unit
mRNA	Messenger RNA
OS	Operating system
QC	Quality Control
RNA	Ribonucleic acid
RNA-SEQ	RNA-Sequencing
XML	Extensible Markup Language

Chapter 1

Introduction

Since the early days of Medicine and Biology, but specially in a recent past, a large number of new techniques and discoveries have been achieved concerning these domains.

The increasing medical and biological knowledge coupled with the improvements in computer's computational power, in other words, the capability of solving complex problems, allowed the development of some techniques, regarding molecular biology, such as RNA-sequencing. Through the use of these new techniques medical care can be enhanced since not only they will allow new treatments and procedures to be successively developed, but also because they will enable the implementation of some medical models within the health system such as the personalized medicine.

1.1 Context

The complex process of obtaining an individual's variation of the human genome, is achieved using a method called RNA-Sequencing, which nowadays represents a method that is gaining traction when the whole-transcriptome profiling is meant to be achieved. This technique comprises several steps and for each one, there are already some tools available, having each of them its own advantages and disadvantages.

The heart of this dissertation is the software called iRAP [FPMB14]. This piece of software implements the pipeline containing every step that makes part of the RNA-Sequencing schema allowing the user to customize it by choosing which of the available tools is going to be used, in each step of the pipeline. The main goal of this Thesis project is then to improve iRAP's performance which is naturally related with the performance of the RNA-Sequencing technique.

The state of the art as far as iRAP software is concerned have been carefully investigated and analyzed aiming at successfully identifying the actual performance of the system and the possible portions of the execution where improvements can be inserted.

1.2 Motivation and Objectives

As expected besides complex, the RNA-Sequencing process is computationally heavy making the whole process to be very time consuming. Not only the operations necessary to sequence genetic information are very weighty but also the whole genome of a human specimen is very large reaching sizes of more than 2900 megabytes in data, it is easily understood the reason why, having to read and process all this data, the pipeline execution is very demanding as far as computational power is concerned.

The main goal of the proposed thesis project is to enhance the presented software, iRAP, responsible for emulating the RNA-Sequencing pipeline. All the improvements to be embedded in the system aim not at improving the results' accuracy but at boosting its processing rates and execution times.

The achievement of this dissertation purpose do represent added values not only for RNA-sequencing techniques but also to medicine in general. The improvements can also encourage a higher number of researches around RNA-Sequencing topic to be taken, since they will allow investigators to put through a larger number of researches spending less time and money than they did before.

Finally this effort in optimizing RNA-sequencing performance through the optimization of iRAP's one, trying to make the whole RNA-Sequencing a less time consuming process, can be seen as one of the first efforts in trying to improve this techniques' performance, which will improve the existing conditions so that new therapies can be developed and to allow new medical models, such as personalized medicine, to be used in the future in common treatments, since in order for this to be used beyond researches the time consumed by the sequencing of the whole human genome must be decreased.

1.3 Dissertation Structure

Besides the current chapter, this dissertation report contains five more chapters aiming at providing a full picture of the existing problem and on the possible solutions to mitigate it. A major purpose of this report, is also to describe the implemented one and its results' evaluation.

In the following Chapter 2 the state-of-the-art review it presented, regarding the alternative methods and pipelines and together with a full analysis that where performed on the iRAP pipeline, which is the heart of this thesis. In Chapter 3 the problem to be solved during this dissertation is going to be specified, as well as possible solutions for mitigating it. In Chapter 4 the details concerning the implementation of the chosen solution are presented. After providing all the details about the implementation process, in the Chapter 5 every test performed, as a mean of validating the developed system, is going to be described as well as its results. Chapter 6 is the last one of this thesis report and aims at presenting not only the main conclusions that can be drawn but also the improvements that can be made in the future.

Chapter 2

State of the Art Review

Since the heart of this thesis project is an already existing software (iRAP), there was the need to understand how it was developed in the first place trying to understand the gap it purposes to suppress, figuring out the motivations that led to its implementation; This way the state-of-the-art was reviewed with care.

The aim of this review is to understand, in the first place, the existing problem as well as the methods that could be used to overcome it. Then, in a following stage taken the analysis of the iRAP itself became the focus of the-state-of-the-art review as a way of pursuing the understanding of, not only it's structure but also the way it operates.

2.1 Knowledge extraction from genetic data

For the last few years, techniques aiming at studying gene expression as well as detecting differential expression genes, or in other words, to detect genes that are only expressed in certain circumstances, have been developed. Since the creation of the first procedure, in the 1990s, the stated techniques rely upon genetic material, which can be either DNA but which is in most cases RNA, in order to catch up the desired results.

The first method developed was Microarrays, which as mentioned before could use both DNA and RNA in order to obtain information about genes' expression. Immediately after its creation, this method allowed some important discoveries to be done regarding some diseases, as the profiling of differential expressed genes in both diseased and healthy tissues or patients allowed the improvement of some cures and medicines. Despite the fact that, as mentioned, right after its creation this technique expressed a lot of potential, it is not perfect and thus comprises several limitations that negatively affect its results. Nonetheless,

> "microarrays remain the most popular approach for transcript profiling and can be readily affordable by many laboratories." [ZFLB+14, chap. Introduction]

More recently a new method have been revealed which is RNA-sequencing. This one is quickly gaining traction, as far as gene expression is concerned, achieving better results than the previous one by trying to bridge the already known constraints of Microarrays. Despite this fact, there are still some drawbacks related to its costs and that is the reason why there are still some limitations to its usage as a method for transcriptome profiling.

2.1.1 Microarrays

One of DNA's properties is the hybridization which is a biological technique that examines the genetic degree of similarity between DNA samples or fragments, the greater the similarity the harder it is to break the connection among both samples. This property represents the core of Microarrays implementation, as it is the most essential step of this procedure.

Microarrays approach comprises several steps in order for gene expression measurements to be possible. Since it relies on genetic information at the first place, it is necessary to gather biological samples form the tissue (s), which are called probes. As previously referred, Microarrays may use both DNA or RNA but in the majority of the experiments the probe will be a mRNA molecule. In the mean time a chip must be created so that each chip's cell contains a well known fragments of genome, the so called targets.

From this step on, hybridization is performed while the chip is flooded with the genetic samples collected at the first stage, so that probes become connected with the targets that were originally present in the chip. The connections established between probes and targets at the described phase represent the genes' expression results, being the connected genes the ones that are expressed.

Owing to this *modus operandi*, the presented technique contains some limitations regarding transcriptome profiling. The first limitation is the fact that this method requires a high degree of before-hand knowledge in order for probes to be collected at the first step, also it demands a lot of genetic information for the analysis to be possible. This drawback represents an obstacle when investigators want to take researches on non-modeled organisms with genome yet to be determined.

There are even flaws related to the main process of the technique, the hybridization. For instance this process is likely to be affected by background noise which will harden the process of detecting whether genes have low expression levels or if they are just not expressed. Lastly in the samples collected, there is the possibility of existing similar probes and if by the time of hybridization they don't agree on each other the analysis of genes' expression can be inaccurate.

2.1.2 RNA-Sequencing

Shortly after its creation, RNA-sequencing started to increasingly gain importance as a method for transcriptome profiling aiming at studying genes' expression. This only became possible due

to the fact that the given technique makes use of recently created deep-sequencing technologies, which are biologically complex, that are also named as next-generation sequencing methods.

Zhao et al. [ZFLB+14] conducted a study which major concern was to compare both RNA-sequencing and Microarrays techniques as respecting to transcriptome profiling. All the tests executed revealed that the results obtained by using both techniques share a high rate of correlation as there is a great amount of overlap between their results. Despite this similarity in results, it was as well recognized that even though a high percentage of genes were detected as differentially expressed by both methodologies, expression profiles, or in other words the expression levels for each gene, were considerably different when comparing both results. It was also noticed that some genes were only detected in one of the techniques as differentially expressed. The stated conclusions evidence that even though both platforms accommodate high correlation in results, there is differences on expression profiles obtained by each of the techniques. Through a more detailed inspection to the differences in the results of both methods, it became clear that RNA-sequencing schema was capable of detecting a broader range of expression values which implies that this method is more clear-cut when detecting genes with very low and very high expression levels, in comparison with Microarrays.

The more accurate results obtained by the RNA-sequencing method are directly related to the flaws existent in Microarrays method as they harm its results, furthermore since both methods don't share the same operational schema these flaws don't affect RNA-sequencing technique allowing it to produce better results. Besides the differences in results obtained, RNA-sequencing still comprises some advantages in relation to Microarrays approach.

> "First, unlike hybridization-based approaches, RNA-Seq is not limited to detecting transcripts that correspond to existing genomic sequence.' [WGS09, chap. Introduction]

This method even provides to genome profiling some new possibilities that were not supported by Microarrays and consistently were not available until RNA-Sequencing creation as Zhao et al. states:

> "RNA-Seq has considerable advantages for examining transcriptome fine structure such as the detection of novel transcripts, allele-specific expression and splice junctions.' [ZFLB+14, chap. Introduction]

Even though RNA-Sequencing has some advantages and even outperforms Microarrays as far as transcriptome profiling results are concerned, this technique also comprises some drawbacks related to its performance due to the more complex computation necessary and the high volume of data that is necessary to store, retrieve and process in order for the RNA-Sequencing pipeline execution to be possible. The stated fact increases execution time and the computational power needed making genes expression analyses more expensive than the it would be using the Microarrays model. This is a major drawback since for large studies it can become really expensive which

is impractical, being this the reason why RNA-Sequencing usage is still being limited and is not in some cases the first choice when a profiling tool is needed. However it is believed by both Wang et al. [WGS09] and Zhao et al. [ZFLB+14] that once this disadvantages are overcome RNA-Seq will replace Microarrays regarding transcriptome analysis.

2.2 iRAP

As mentioned before along the analysis of the techniques available for retrieving knowledge from genetic data, in the previous subsection, RNA-sequencing is a complex mechanism for transcriptome profiling which due to the excellent results it is able to achieve is gaining popularity as a method for genome sequencing.

This technique comprises several phases that are going to be further analyzed. For each of these steps there were already, before iRAP's creation, several tools which implement them, however the execution of the sequencing process was very hard to achieve due to the numerous tools required and the differences between the formats of the outputs and the inputs required by the following tools, hardening the process of integration the whole set of tools chosen for the genome sequencing execution.

iRAP, which is the fundamental piece of this thesis project, then came to light as a free software trying to solve the presented difficulties through the implementation of the whole RNA-sequencing pipeline. This software relies upon the already existing tools for each step of the sequencing process masking the execution of a tool for each phase of the pipeline, allowing the end user to execute only one program with the configuration desired, giving him the possibility to choose among every existing tools, for each step, the one to be used. As every step of the execution is automated by this software, at end of the sequencing process iRAP generates specific web reports for each phase in order for the results to be easily evaluated.

In the Figure 2.1 bellow, a representation of RNA-sequencing pipeline is presented, in which every step is well distinguishable.

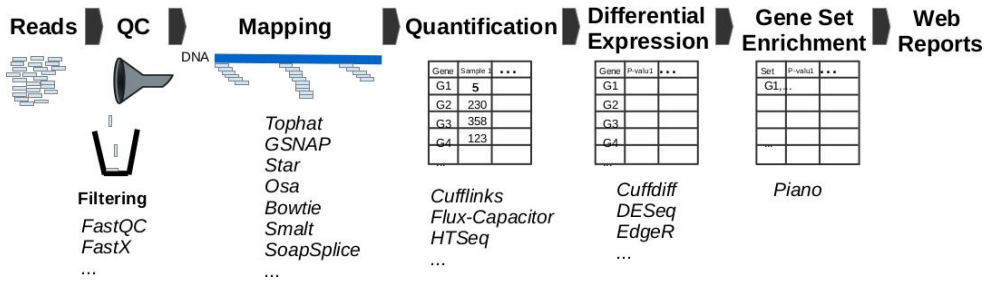

Figure 2.1: Pipeline Representation

Like what can be seen in Figure 2.1, sequencing process implemented by iRAP is divided in six different steps, respectively in the following order.

- **QC** — Quality control.
- **Mapping**
- **Quantification**
- **Differential Expression**
- **Gene Set Enrichment**
- **Web Reports**

However despite the steps exhibited in the representation, not all this steps make part of the RNA-Sequencing process, as some of these are just extra features provided by the pipeline software.

The first step in the implemented pipeline (Quality Control) represents a phase that is not directly contained in the RNA-Sequencing technique which can be optionally included in iRAP's execution. At this first stage of execution, a quality-based filtering is performed over the reads passed as an input to the program. The filtering process mentioned can be divided in three different chapters according to the different type of filtering fulfilled in each one of them. For starters the first filtering process is based on the quality of the reads, a threshold is defined representing the minimum quality value that must be accepted, and every base at each read with a lower quality value than the lower limit is automatically eliminated. Still in this first step, after the performed filtering, the reads length is evaluated in order to gather information about their length, to test whether their size fall behind the minimum imposed. The second stage is responsible for the filtering process based on the possible contamination that may occur while acquiring the reads, for this process every read is assessed to measure the possibility of that specific read to be part of another specie rather than the one that is being analyzed, in this case those reads would immediately be discarded. The final reads restriction is done by elimination undefined bases in every read available.

The Mapping or alignment is the second step in the iRAP's execution representing the first one in RNA-Sequencing process. At this stage the already filtered reads must be aligned to the species genome.

> "A critical step in RNA-Seq data analysis is the alignment of partial transcript reads to a reference genome sequence.' [ESS+14, chap. Abstract]

For this second step of the program execution the end user of iRAP software can configure it to use one of the possible mappers: *Tophat1* [TPS09], *Tophat2* [KPT+13], *Osa* [HGNL12], *Star* [DDS+13], *GSNAP* [WN10], *Bowtie1* [LTPS09], *Bowtie2* [LS12], *Smalt*, *BWA1* [LD09], *BWA2* [LD10], *GEM* and *SOAPsplice* [HZL+11].

After the mapper execution, a quantifier method is needed concerning, as the name implies, the quantification of every entity with biological meaning. Again likewise what happen with the previous step of the pipeline, the user have the ability to choose which of the available tools, for calculating the abundance of each gene in each sample provided, he wants to use. This choice may, for example, vary from the requirements and goals of the experiment underway. For this specific stage, iRAP allows the use of the following tools: *HTSeq* [APH15], *Cufflinks* [TWP+11], *NURD* [MZ13] and finally *Flux-capacitator*.

Subsequently to the quantifier execution a differential expression analysis can be performed using the results originated from the quantification tool. This study aims at finding genes that exhibit different expression levels on distinct biological scenarios, identifying genes that can either be, or not, expressed under different circumstances.

> "The field of differential expression analysis of RNA-seq data is still in its infancy and new methods are continuously being presented.' [SD13, chap. Background]

As stated differential expression analysis is still limited to simple designs as it is a recent technology in constant development. However, it already holds a lot of importance among transcriptome profiling studies since the most common use for them it to recognize deferentially expressed genes. For this stage of execution the tools available are: *Cuffdiff1*, *Cuffdiff2*, *DEseq* and *EdgeR*.

Regarding the 5th step of iRAP's pipeline, Gene Set Enrichment, only one tool is available to be used in this software, the *Piano R*. This stage works as a complementary one to the differential expression genes analysis, since the last one ignores important data that can be beneficial to biological studies. Gene Set Enrichment besides the gene's overall expression inspection performed by the previous step, do as well evaluate alternative splicing.

The last node of the execution does not belong to RNA-Sequencing technique, and represents only a feature provided by iRAP in order to ease the process of scrutinizing the results obtained at each step of the pipeline. For each one of them, web reports are generated with the summary of each tool's execution. Since every step contained in the pipeline comprises very complex processes that requires a lot of computation to be done, iRAP during the execution keeps track of temporary files with the pipeline progress so that if the execution, for some reason, is stopped, when re-executing it the whole computation is not required to be done again.

iRAP is a software that, as specified, implements the whole pipeline necessary for RNA-Sequencing process. As expected every handicap associated with this technique is automatically broadcast to the presented software, this way the massive amount of data to be load, processed and stored continues to represent a drawback to the usage of RNA-Seq approach and naturally to the use of iRAP. In the next subsection, some software alternatives to iRAP, which as well implement the whole pipeline allowing genome sequencing processes to be performed, will be discussed. However the mentioned drawbacks associated with the RNA-Sequencing technique are also valid for them.

2.3 Alternative pipelines

In this subsection three alternative software to iRAP are going to be approached. Even though every one of them represents a different implementation to the sequencing process pipeline, they all share among each other and with iRAP the same main goal, the same basic stages, so that the genome sequencing can be successfully achieved, and the same basic drawbacks associated with RNA-Sequencing technique.

Even though every pipeline that will be presented comprises the same primary steps necessary for genome sequencing they implement / allow the usage of different tools at each one of those stages. Regarding Grape RNA-Sequencing pipeline [KRMG13], at the Mapping phase this software only let on the usage of *GEM*. At the end of the mapping stage the following one is the Quantification process, at this step Grape uses *FluxCapacitator*, using two different approaches in order to calculate the abundance of every biological unit. Finally, after the quantification process, in order to determine differential expressed genes, Grape resort to *Cufflinks* tool to achieve it.

In contrast to what happens with Grape pipeline, HTS [GTBK11] allows more than one tool to be used at each step giving the end user the option to choose which one is the most convenient for the experiment or study to be performed. For HTS pipeline, at the mapping stage, there are three possibilities of mappers that can be used: *BWA*, *Bowtie* and *Tophat*. Once the aligning process is finished, for the quantification step *Cufflinks* or *MMSeq* can be used, however at this step, pipeline reaches its ending state, as there is no method supported for differential expression analysis.

Finally Galaxy [GNT10] is a web-based software that, as well as the others, implements the pipeline for genome sequencing. This platform basic principle is that a tool for this purpose should be as accessible and as easy to use as possible, trying to reduce some difficulties felt by scientists when a transcriptome profiling experiment is being performed. Galaxy tool, like Grape, only allows one tool to be used at each step of the pipeline, for the mapping stage *Tophat* is used followed by *Cufflinks* for quantification and *Cuffdiff* for the execution of differential expression analysis.

Having in mind the analysis performed on the existing programs to implement RNA-Sequencing process, with every step contained in it, it is easily realized that one of the biggest advantages of Galaxy web-based platform is the fact that in the process of the platform development a great attention has been given to its usability trying to make it as simple to use as possible, which will, upon an experiment or a study, be a great benefit for the responsible researchers. However, despite this advantage it is clear that this platform is one of the tools that allows less methods to be used at each step, not allowing any customization of the pipeline, since only a tool can be used for each step. Sharing this drawback also Grape pipeline does not provide customization options. When comparing iRAP to the three alternatives addressed in this subsection, iRAP grants the end user with a much higher degree of freedom when configuring the pipeline to be executed, since iRAP allows, at each step, a lot more tools to be chosen by the user. This fact is not only good due to the great amount of configuration options possible for the pipeline execution but also because

different experiences and studies on genome sequencing may require different tools to be used at each stage of the pipeline, which will not be possible using the other pipeline's implementations. The advantages of iRAP's usage against the other implementations represents the main reason for iRAP to be the heart of the thesis project.

2.4 Mapping and Quantification

Both mapping and quantification are processes that make part of the RNA-Sequencing pipeline, corresponding in iRAP pipeline to the second and third step respectively.

> "Accurately quantifying gene expression levels is a key goal of experiments using RNA-Sequencing to assay the transcriptome. This typically requires aligning the short reads generated to the genome or transcriptome before quantifying expression of pre-defined sets of genes.' [FMB14, chap. Abstract]

Just like expected this two stages have a major importance in the experiments results being them either the quantification of genes or the analysis of differential expression genes under certain or different biological contexts. In iRAP's pipeline, there is the possibility for the user to customize it, deciding the tools to be used during the execution of the sequencing processes, however, even though it is obvious the importance of mapping and quantification to the final result, the consequences of choosing the right / best tool, or not, to execute them are not as straight forward.

Having the main goal of putting this matter to the test, Engström et al. [ESS+14] conducted a study with the purpose of evaluating the existing and available alignment programs' accuracy, as far as alignment results are concerned. Considering that, once finished the mapping step, the results achieved by the chosen tool will represent the starting point for the rest of the pipeline execution, more accurate results at this step should provide rigorous final results. Through the tests performed with every mapper available, it was evident that different accuracy have been obtained for each one of them, with some of them clearly outperforming the others, achieving more correctly aligned reads and less mismatches, having *GSNAP*, *MAPsplice* and *STAR* collected the best results among the whole set of mappers. Despite this conclusions, the execution has been restarted and the quantification method (*Cufflinks*) have been executed in order to obtain the quantification results, after the term of *Cufflinks*' execution the results with every mapper have been compared and the experiment made possible to perceive that after the execution of the quantification method, the choice done for the mapper tool did not influence the final result that much since the following mappers revealed, at the end, similar results: *GEM*, *GSNAP*, *MAPsplice*, *STAR* and *Tophat*.

Later to this study another one performed by Fonseca et al. [FMB14] had the goal of understanding if different pipelines would affect gene expression levels obtained from the same input data. In this research both alignment and quantification stages have been executed by a various

number of tools in order to measure the results collected by different pipelines configurations with different combinations of mappers and quantifiers. A large number of combinations between mapper and quantifier have been executed which allowed, for each single quantifier to test pipelines with different mappers. One conclusion was derived from the obtained results, which was the fact that some quantifier methods are less dependent on the mapper choice, as the result's variation using different mappers for the same quantifier were significantly lower for some quantifiers than the variation for the others.

In the figure 2.2 bellow, the results obtained in the executed tests during the study [FMB14] are presented. Two methods were used to compute the ranking for each combination, having these been coupled into an "Overall Rank" easing the interpretation of the outcomes. Through the analysis of the results it can clearly be seen that, and as stated by the other study presented [ESS+14], the choice of the mapper doesn't have a great amount of influence in the final result, since pipelines with the same quantifier and different alignment tools obtain similar results.

Pipeline		Overall	Error		Spearman	
Aligner	Quant. Method	Rank	Avg. Rank	mean ± sd	Avg. Rank	mean ± sd
osa	htseq-ine	17	9	16.88 ± 2.00	8	0.94 ± 0.01
tophat1	htseq-ine	18	10	17.28 ± 2.52	8	0.94 ± 0.01
gsnap	htseq-ine	23	11	19.87 ± 7.78	12	0.93 ± 0.01
tophat2	htseq-ine	23	13	19.10 ± 5.93	10	0.94 ± 0.01
osa	fluxcapacitor	24	22	19.95 ± 2.04	2	0.95 ± 0.00
star	htseq-ine	24	11	17.37 ± 2.81	13	0.93 ± 0.01
smalt	htseq-ine	25	14	18.93 ± 7.22	11	0.93 ± 0.00
tophat1	fluxcapacitor	27	23	19.79 ± 2.08	4	0.94 ± 0.00
bwa2	htseq-ine	28	16	20.34 ± 6.05	12	0.93 ± 0.02
star	fluxcapacitor	29	23	20.51 ± 2.72	6	0.94 ± 0.00
tophat1	cufflinks2	31	13	22.74 ± 16.44	18	0.92 ± 0.03
osa	cufflinks2	33	13	20.87 ± 9.57	20	0.92 ± 0.03
star	cufflinks2	34	12	16.94 ± 2.96	22	0.92 ± 0.01
tophat2	fluxcapacitor	34	26	22.17 ± 4.44	9	0.94 ± 0.01
bwa2	fluxcapacitor	36	26	20.84 ± 2.99	10	0.94 ± 0.01
osa	htseq-u	36	16	22.07 ± 15.13	21	0.92 ± 0.02
tophat1	cufflinks1	36	13	22.35 ± 15.58	23	0.92 ± 0.01
tophat1	htseq-u	36	17	19.84 ± 4.83	19	0.92 ± 0.00
bwa2	htseq-u	37	20	23.75 ± 13.71	17	0.92 ± 0.02
gsnap	cufflinks2	37	16	24.26 ± 17.44	22	0.91 ± 0.05
gsnap	fluxcapacitor	37	26	22.71 ± 6.43	11	0.94 ± 0.00
smalt	htseq-u	37	19	20.30 ± 7.67	19	0.92 ± 0.01
osa	cufflinks1	38	13	21.12 ± 12.17	25	0.92 ± 0.01
bwa1	htseq-ine	39	21	23.84 ± 8.04	18	0.92 ± 0.03
star	cufflinks1	39	12	17.65 ± 4.28	28	0.91 ± 0.01
tophat2	cufflinks2	39	14	23.35 ± 16.97	25	0.91 ± 0.07
gsnap	htseq-u	40	16	23.46 ± 13.44	24	0.92 ± 0.01
tophat2	htseq-u	40	19	21.44 ± 7.53	21	0.92 ± 0.01
bwa2	cufflinks2	42	22	25.93 ± 15.93	19	0.91 ± 0.05
star	htseq-u	42	18	18.48 ± 4.69	24	0.92 ± 0.00
gsnap	cufflinks1	44	17	25.40 ± 17.08	27	0.91 ± 0.04
bwa1	htseq-u	48	26	26.14 ± 8.57	22	0.91 ± 0.03
tophat2	cufflinks1	48	18	27.41 ± 21.56	30	0.90 ± 0.07
smalt	cufflinks2	50	24	25.23 ± 15.73	26	0.90 ± 0.04
bwa2	cufflinks1	52	26	33.95 ± 22.11	27	0.88 ± 0.08
smalt	cufflinks1	52	25	26.74 ± 16.89	28	0.90 ± 0.06
smalt	fluxcapacitor	52	28	26.59 ± 8.91	24	0.91 ± 0.03
bwa1	cufflinks2	53	27	31.39 ± 17.84	26	0.88 ± 0.07
bwa1	fluxcapacitor	54	31	30.97 ± 11.13	23	0.89 ± 0.05
bwa1	cufflinks1	55	27	36.30 ± 22.90	28	0.87 ± 0.08
bowtie2	htseq-ine	64	29	29.67 ± 12.57	35	0.86 ± 0.03
bowtie1	fluxcapacitor	66	34	34.20 ± 11.43	32	0.87 ± 0.04
bowtie2	htseq-u	68	32	32.96 ± 15.04	37	0.85 ± 0.03
bowtie2	fluxcapacitor	72	35	34.49 ± 8.45	37	0.83 ± 0.05
bowtie2	cufflinks2	76	37	43.69 ± 21.38	39	0.82 ± 0.07
bowtie2	cufflinks1	77	36	40.13 ± 18.6	40	0.81 ± 0.05

Figure 2.2: Pipelines rankings

As stated by Fonseca et al. [FMB14], in the performed study, it is recognized by looking at the tests' results presented in the figure 2.2, that the the overall pipeline ranking seems to group the pipelines' configuration by the quantification method used. Being so, influencing the overall quality the most it suggests that the selection of the quantification method is more relevant than the aligner being used.

Lastly it is even possible to conclude that among every tested pipeline the ones using as quantifier the following methods, were the ones that obtained better overall results: *HTSeq*, *Fluxcapacitator* and *Cufflinks2*.

2.5 Differential Expression

Likewise what have been observed in the previous stages of the pipeline execution, also in the differential expression step the user is allowed to choose the desired tool among those provided by iRAP. This phase represents as well a crucial one for the pipeline execution since most studies nowadays aim at finding differential expression genes between different biological circumstances. However, once again, there was no information about the effect, on the results, of choosing one tool over the others. Aiming at figuring out the effect each tool would inflict on the differential expression analysis, Soneson et al. [SD13] conducted a comparison of eleven different methods using simulated data.

From the presented study several conclusions could be extracted through a careful observation of the obtained results for every single available tool. For instance it is possible to assure that some tools outperformed the others for large sample sizes and that other performed better for experiments with reduced sample sizes. Despite this fact it was clearly noticeable that for the generality of the experiments both *DESeq* and *edgeR* were part of the set of tools that achieved the best results. However the most important conclusion to be elapsed from the obtained outcomes is that, yet, it is not possible to agree on the best tools available for differential expression studies, since the results provided by each one of the methods vary with the experiment itself.

Being so, and as believed by Soneson et al. [SD13], among the evaluated methods no one transcended the others, revealing that none of them is optimal under all circumstances. This way, regarding the choice of a differential expression method in a particular situation, there is no right choice to make since the results provided by every tool fluctuate and depends on the experimental conditions.

2.6 Conclusions

By the end of this State of the Art review important knowledge have been acquired regarding possible alternatives to the techniques and technologies that represent the spotlight of this thesis project. An investigation on Microarrays and RNA-Sequencing methods have been done, allowing the understanding about which is the most used technique, the flaws of each one of them and the advantages of the more recently developed one. Lately iRAP became the focus of this review aiming at better perceiving the way it works and the stages contained in its execution. Since there are as well other programs offering the capability of executing RNA-Sequencing process, they were as well analyzed in order to understand the main differences between iRAP and the other existing methods; From this analysis it became clear that iRAP allows a higher degree of freedom to the end user since this one can customize the pipeline a lot more in comparison to the other alternative programs. As mentioned iRAP allows, at each stage of the pipeline, different tools to be used, being so it was crucial to understand the importance of this choices for the final result of the sequencing process. By the research done it became possible to understand that, in the first place, the

different mappers available had in fact different results but the variations it causes in the ultimate results obtained were not so much significant, having the quantification much more expression over the outcome of the whole process. Finally the influence of differential expression tools have, as well, been evaluated however, despite the obvious influence on differential expression results, no significant conclusion could be derived about the best tools to choose since it turned out to depend a lot on the experience context.

Chapter 3

Problem characterization and possible solutions

The basis of this thesis project is the already existing software iRAP, which aims at implementing a biological technique called RNA-Sequencing which have been previously characterized. Despite the great results the biological technique for genes quantification and differential expression genes analysis is able to achieve, outperforming a well known technique for the same purpose, the Microarrays, it also implies some drawbacks associated with genome sequencing.

iRAP as a tool for RNA-Sequencing suffers inevitably from performance issues fact that have been and still keep restraining the genome sequencing from becoming a more used method, even though it achieves the best results when compared with Microarrays alternative. It is easily explainable the reason why this handicap exists, for starters the pipeline responsible for genome sequencing contains several stages, being so that each one of them requires great amount of computational power to execute, as complex biological operations need to be done at each step. Moreover the data to be gathered, processed and stored is of huge size, for example, for humans, the genome can reach sizes of more than 2900 megabytes of data, which associated with the complex processes to be executed ends up converting the iRAP pipeline, or any other RNA-sequencing pipeline, in a very time consuming sequence of processes. Coupled with the performance problem is the financial one, as the lack of performance in the pipeline represents a direct and awful impact on the experiences' and studies' costs, considering that this way it requires more time and more computational resources for the results to be achieved.

As stated by Zhao et al. [ZFLB+14] RNA-Seq is a powerful technology that is expected, in the future, to replace Microarrays regarding transcriptome profiling. Despite the better results achieved, when compared to the Microarrays alternative, the challenges associated with RNA-Seq are currently limiting its potential use since the higher costs of the RNA-Seq technique make it impractical for large studies.

The optimization of this RNA-Sequencing pipeline, in order to allow an increasing amount of studies to use this technique, is then crucial, which in fact is the main goal of this dissertation project. iRAP have been the chosen software, since concerning genome sequencing pipelines this is the one which provides more features and also an extended set of pipeline's customization for the end user to benefit from, when an experiment or a study is being performed.

3.1 Possible Solutions

After a profound analysis of the RNA-Sequencing tool in study (iRAP), two possible solutions are believed to be adequate as a mean of mitigating, or at least attenuating the previously scrutinized problem.

The first and undeniably obvious approach is the optimization of the pipeline, using parallelization techniques to do so, trying to get the most out of the available hardware that nowadays is built based on multicore architecture schema. However the goal of enhancing the whole pipeline, knowing by hand the great amount of tools at each stage is far too greedy. This way the main objective needed to be further specified by founding a more doable and realistic goal.

Despite the fact that the presented alternative is the one which can be more easily translated in direct improvements on the pipeline's performance, there is even the possibility of implementing a scheduler, which will associated with cluster composed of multiple computers or processors, after the decomposition of the whole pipeline execution into separate tasks holding their dependencies, provide the best scheduling possible allowing the tasks to be executed in the optimum order and at the ideal time and computer, taking into account the executions costs associated with each one of them.

3.1.1 Performance Enhancement through Parallelization

As mentioned before, regarding the optimization of RNA-Sequencing processes for resorting to the parallelization of the pipeline in study, there was the need to restrict the problem context, with the aim of accomplishing it, the iRAP pipeline had been fully analyzed and tested searching for the most time consuming phases. The results of the executed tests had later been analyzed and can be looked at, in the tables bellow.

Table 3.1: Executed tests for iRAP profiling

Id	Mapper	Quantifier	Differential Expression
1	Tophat1	HTSeq1	DEseq
2	Tophat2	HTSeq2	DEseq2
3	GSNAP	Cufflinks1	Cuffdiff1
4	Tophat2	Cufflinks2	Cuffdiff2
5	Tophat2	Cufflinks1_nd	DEseq2
6	Tophat2	Cufflinks2_nd	DEseq2
7	BWA2	——	——
8	Bowtie2	HTSeq2	DEseq2
9	GEM	NURD	DEseq2
10	Tophat2	HTSeq2	EdgeR
11	GSNAP	HTSeq2	DEseq2
12	Bowtie1	HTSeq2	DEseq2
13	Tophat2	Cufflinks1	Cuffdiff1
14	Tophat1	Cufflinks1	Cuffdiff1
15	GEM	Cufflinks1	Cuffdiff1
16	Bowtie2	Cufflinks1	Cuffdiff1
17	Tophat1	NURD	DEseq
18	GEM	Cufflinks2	Cuffdiff2
19	Tophat1	Cufflinks2	DEseq
20	Tophat2	Cufflinks2	Cuffdiff1
21	GSNAP	HTSeq1	EdgeR
22	Bowtie1	HTSeq1	EdgeR
23	GEM	Cufflinks1_nd	DEseq
24	GEM	Cufflinks2_nd	DEseq
25	Tophat1	Cufflinks2	Cuffdiff2
26	Tophat2	Cufflinks2	Cuffdiff2
27	Tophat1	Cufflinks1	Cuffdiff1
28	Tophat2	Cufflinks1	Cuffdiff1
29	GEM	Cufflinks1	Cuffdiff1
30	GEM	Cufflinks2	Cuffdiff2
31	Bowtie1	Cufflinks1	Cuffdiff1
32	Bowtie2	Cufflinks1	Cuffdiff1
33	Bowtie1	Cufflinks2	Cuffdiff2
34	Bowtie2	Cufflinks2	Cuffdiff2
35	GSNAP	Cufflinks2	Cuffdiff2

In the table 3.1 presented above, it is possible to see every test that have been performed on iRAP pipeline. Not every possible combination of tools had been taken into consideration since there are incompatible tools, restricting the number of possible combinations of methods. Besides this, there are even some tools that could not be used since the test data did not support them. At the end of the testing phase some tests have only been executed in order to better evaluate some tools' performance, since previously their behavior had already pointed that specific tool as a possible sub-problem, representing this way the so wanted reasonable goal.

Table 3.2: Time elapsed during the tests for iRAP profiling

Id	Filtering	Alignment	Quantification	Differential Expression
1	14 sec	9min 16sec	6min 21sec	22 sec
2	19 sec	10min 23sec	5min 51sec	32 sec
3	2min 58sec	3min 22sec	50h 52min 49sec	1min 17sec
4	5min 30sec	9min 48sec	9min 56sec	22min 0sec
5	5min 40sec	9min 48sec	1min 17sec	32 sec
6	5min 38sec	9min 49sec	1min 59sec	31 sec
7	5min 26sec	14min 18sec	—-	—-
8	4min 52sec	4min 13sec	5min 39sec	29 sec
9	5min 36sec	9min 42sec	51 sec	30 sec
10	4min 56sec	10min 14sec	5min 42sec	12 sec
11	6min 34sec	2min 58sec	5min 14sec	30 sec
12	4min 49sec	3min 53sec	3min 28sec	—-
13	5min 25sec	9min 54sec	9min 18sec	14min 5sec
14	5min 30sec	10min 5sec	33min 48sec	33min 37sec
15	5min 30sec	9min 26sec	3min 50sec	33min 15sec
16	5min 30sec	4min 10sec	3min 54sec	34min 37sec
17	5min 47sec	10min 22sec	49 sec	23 sec
18	5min 46sec	10min 13sec	4min 54sec	53min 27sec
19	5min 41sec	10min 34sec	37min 15sec	33 sec
20	5min 46sec	10min 18sec	10min 53sec	18min 5sec
21	7min 0sec	3min 22sec	5min 17sec	11 sec
22	5min 5sec	4min 6sec	3min 28sec	—-
23	5min 57sec	10min 18sec	1min 21sec	28 sec
24	5min 21sec	9min 9sec	1min 52sec	26 sec
25	5min 28sec	9min 49sec	34min 43sec	48min 7sec
26	5min 41sec	9min 41sec	9min 54sec	21min 8sec
27	5min 39sec	9min 53sec	33min 23sec	27min 44sec
28	5min 34sec	9min 48sec	9min 16sec	13min 15sec
29	5min 40sec	9min 40sec	3min 24sec	27min 46sec
30	5min 39sec	9min 24sec	4min 12sec	43min 35sec
31	5min 35sec	3min 35sec	2min 6sec	—-
32	5min 27sec	3min 49sec	3min 37sec	28min 56sec
33	5min 39sec	3min 42sec	3min 0sec	—-
34	5min 31sec	3min 52sec	4min 26sec	48min 45sec
35	7min 6sec	2min 50sec	47h 16min 57sec	6min 36sec

In the previous table 3.2, for each executed test presented in the table 3.1, the duration in time of each stage is presented, aiming at identifying the most time consuming phases or tools within the iRAP pipeline, which provides not only a better general idea about the behavior of the pipeline and the tools in it contained, but also a feeling about where to focus the optimization processes among the pipeline.

The first step of the presented pipeline is the Filtering one, this process aims at filtering the reads that the program receives as input. This filtering step is only performed if the user desires so,

being executed by *FastQC* tool, this way the possibility to choose the tool to execute it, does not exist as it does in other pipeline's stages. As so it is expected that, for the same reads, the amount of time required to fulfill the whole process to be very alike among all the executions done. In fact this can be checked by looking at the table 3.2 under the column "Filtering". By looking at it, if the first two executions are repudiated, ts becomes clear that even with some little variations the time elapsed is always similar. For this matter the first two tests executed are being dismissed as, the very different values obtained, may be explained because probably the temporary files that iRAP creates, for not repeating operations, have not been properly deleted before the test's execution. Since in the majority of the tests this stage does not represent the major slice, as time is concerned, of the pipeline execution it does not assume the role of the best phase where to focus the attention, regarding the pipeline optimization.

At the second stage of the pipeline the reads are aligned with the species genome, for this several tools can be used as described earlier, at the state-of-the-art. It is expected then, that using different tools for the mapping process the computational time needed presents variation from one tool to the others. In fact by looking at the Alignment columns this can be verified, as some tools in fact take considerably more time to execute than others, in some cases even representing the biggest percentage of the total execution time for the correspondent test, in comparison to the tools used on the other stages of the pipeline. This is a fact of big importance since it could represent a possible problem to be solved by becoming the target of this thesis project.

For the quantification process, like what have been done for the alignment and differential expression analysis, different methods have been tested and evaluated. From the study of the obtained computational times elapsed for each test, at this stage of the pipeline, it becomes pretty evident that serious fluctuations exists between the available quantifiers. The main reason for this huge variations on the execution time is due to the fact that *Cufflinks* methods at some tests, concerning execution time, performs a lot worse than the other existing quantification tools. It is even noticeable that in some executions, this stage is the one with most negative expression in the total execution time of the pipeline and that, even in some cases, the time consumed by the *Cufflinks* methods is so high that it represents almost one hundred percent of the total amount of time required for the whole pipeline to execute. Being so *Cufflinks* represents a sub-problem of the pipeline that could easily be the target of this dissertation.

Like what have been done with the other stages of the pipeline so did the differential expression analysis been as well evaluated using different tools available for the purpose. Despite the fact that some tools at this phase almost consume no time when comparing to the rest of the pipeline execution, it can be observed that the *Cuffdiff* methods requires a much higher execution time. In some cases this stage represents the most time consuming one which indicates a possible problem to be solved in this thesis context.

The fundamental goal of this dissertation is to optimize the iRAP pipeline which is a tool for executing RNA-Sequencing technique. However, not only this goal represents a very scattered

goal but also it is too ambitious for the thesis context. Being so the objectives of this dissertation needed to be restrained for their fulfillment to be possible to achieve. Either in the Alignment, Quantification or Differential expression analysis there are, as previously evaluated, possible problems that could be adopted to become the main target of this thesis project. However just one of this three problems should be chosen to be the focus of this project, in order to specify a concise problem that can, in fact, be solved in the dissertation context. From the identified issues the one that seemed to have more importance is the one related with the quantification stage of the pipeline, accordingly the inefficiency of *Cufflinks* tool as a method of quantification has been decided to be the main problem to solve in this thesis, being so this tool will be object of intense optimization aiming at improving its performance and necessarily iRAP's performance as well. There are many reasons why the problem chosen to be the target of this dissertation overrode the other problems identified through the analysis of the executed tests. First of all, from the examination of the execution times for each step, it is easy to understand that in some executions the time needed for quantification process, using *Cufflinks* tools, becomes a lot superior when in comparison to the alignment and differential expression stages; While *Cufflinks* can take up to more than 50 Hours to execute, alignment and differential expression processes never took more than, 15 and 54 minutes respectively. The problem issued in the Mapping stage, even lost magnitude for two more reasons; Firstly as mentioned every tool available requires very different execution times which implies that no tool is falling behind the others as far as performance is concerned; The second reason relies on the fact that, as suggested by previously presented studies, the choice of the method to execute the alignment process does not influence the end result, of the genome sequencing process, as much since this is much more affected by the choice of the quantifier to be used. As far as Differential expression stage is concerned, also here a big variation was noticed between *Cuffdiff* methods and all the other ones available, however, and as already mentioned, some studies have been performed in order to find out the importance of choosing the best tool for this phase, although no significant conclusion could be taken, since the best tool would depend from the experiment itself. Being so, since there are several tools that could provide equivalent results spending less time, focusing this thesis project on this stage would be not so useful for the optimization of the pipeline.

If this option gets to be the one chosen, as the most promising one as far as performance improvements are concerned, as mentioned above and supported by all the studies analyzed and all the tests executed over iRAP pipeline, *Cufflinks* tool which is available in the Quantification stage of the pipeline's execution is going to be the focus of this thesis. Since computational execution time needed for executing *Cufflinks* can become pretty massive in some pipelines, the main goal of this dissertation would be to improving it, concerning its performance, which will automatically optimize iRAP and naturally RNA-Sequencing technique. Even though *Cufflinks* get to be the focus of the optimization process, also *Cuffdiff* will, as a secondary objective, if at the end of *Cufflinks*' enhancement, time allows so.

For the enhancement of the quantification tool to be possible several steps need to be performed aiming at achieving a considerable speedup in the execution of *Cufflinks*. A step of major importance is the analysis of every algorithm contained and implemented by this tool, this analysis will reveal not only, for each of the algorithms, its parallelization potential, whether or not a specific algorithm can be parallelized, but also the amount of computation being done and the amount of read and write operations. This knowledge is a matter of great importance since, even though an algorithm is able to be parallelized, there is the need to understand for each one of them whether it should be so, using multicore CPU's or using GPU's which are a lot more powerful, being able to execute more complex tasks in much less time than a CPU would take. However the data transfer speed, for applications where the computation is being done on the GPU is much lower than for CPU. Gregg et al. conducted a study [GH11] that aimed at comparing the data transfer speed for some applications when being executed by both the computer's CPU and GPU. For application depending on large data sets, it have been calculated that the memory-transfer overhead combined with the kernel time took longer than 50x the GPU processing time itself, which is the reason why read and write operations within each algorithm needs to be taken into account.

Another major step that is going to be performed concerning the *Cufflinks* optimization is to improve the way data in memory is being accessed so that the algorithms make the best use possible of the cache memory available, which is the fastest existing memory, reducing as much as possible the need from accessing the RAM memory. Through the parallelization of every possible algorithm comprised in *Cufflinks*, using multicore CPU's or even GPU's and the improvement on memory access a significant speedup is expected to be achieved at the end of this dissertation project. At a final stage, if this approach gets to be the chosen one to implement, the obtained results are going to me measured in two ways, firstly by evaluating the improvements on *Cufflinks*' processing rate and through the use of iRAP's test data, executions times are, once again, going to be measured in order to quantify the obtained optimization.

3.1.2 Scheduler

The second approach, that represents one of the options that can be taken in order to reach the desired goals of this thesis project and which have been previously mentioned, consists on the implementation of a scheduler which gives ideally the optimum order and time at which each of the tasks, that make part on the pipeline execution, must be invoked.

Even though schedulers always seem to be a very promising approach to take when performance's improvement is the main goal to accomplish, as their main purpose is to diminish the total time or *makespan* needed to complete the execution of a specific set of tasks; As far as RNA-Sequencing is concerned, through the use of iRAP, the possible gains seem, at the first glance, to be very disappointing when in comparison to the ones that sound possible to obtain by introducing parallelization. There are two main reasons for this feel of dissatisfaction towards the benefits that can be achieved with the implementation of the scheduler. For starters considering the execution of a configuration of one of the possible iRAP pipelines, the schema of tasks' execution that would

result of the scheduling process, would inevitably be indistinguishable from the normal flow of execution considering the order at which the existing tasks are already being executed in the current pipeline (iRAP), this results from the fact that only one tasks is being executed at each stage of the pipeline, and that it necessarily depends from the previous stages. Furthermore, even forgetting about the dependencies schema between the stages of the pipeline, that was just explained, being executed in a single computer regardless of the schedule provided by the scheduler the total *makespan* that the whole set of tasks would take would not change since the set of tasks would be the same with every single task needing the same time as before to execute.

Regardless of being apparently not suitable to be used as a mean to reach the main goal of this thesis, a scheduler can actually introduce improvements not in the performance of the pipeline, since each task itself will not be any faster to execute, as explained before, but in the reduction of the overall time stamp, getting the most out of the existing pipeline. However this decrease in the pipeline's execution time cannot be achieved only by the implementation of a scheduler, since as mentioned before if all iRAP's fundamentals are maintained the scheduling output would be exactly the same as the normal flow of the pipeline. Having this in mind it is easily understood that, in order to achieve the objectives of this thesis project, a control module must be developed with the major role of executing the right tasks at the right time, accordingly to the scheduling output, although this is not enough since the scheduling, as explained, was going to be equal to the existing execution flow. Taking this into account, to be possible to take the most out of these two modules, both the control one and the scheduler must support distributing computing, allowing the execution of the pipeline's set of tasks to be done by not only a computer / node, but by a cluster of them.

Nevertheless, even though the stated above represents a great increase in the computational power of the whole system executing the pipeline responsible for the RNA-Sequencing, even when allied with the scheduling output provided by the scheduler it wouldn't be able able to reduce the time needed for the whole pipeline execution meaning no improvement at all would be achieved. While offbeat this would take place due to the fact that each task, representing each stage of the pipeline, would be waiting for the completion of the task that compose the phase immediately before, resulting only in a lot of misused computational power. Although if the control module is implemented allowing the execution of multiple pipelines at the same time, this would make possible for different pipelines to be executed in parallel, in this scenario the scheduler would map each task to be executed in the most suitable nod of the cluster having into consideration its dependencies and costs of being executed in each one of the available computers trying to extract the most out of the available hardware while trying to reduce the *makespan* as much as possible. Another feature that if allowed by the control module could improve both the execution time indirectly and the results accuracy is the acceptance of multiple equal tasks to be performed as well as a task performing as a voter to joint their results aiming at achieving a better outcome as far as quality and accuracy are concerned.

The implementation of a scheduler, aiming at providing the best execution schema for the set

of tasks in the need of being executed, coupled with the possibility of distributing the computation necessary to execute them among a cluster of computers or processors besides the already stated benefits, provides new possibilities for improvements to be achieved that were not possible to obtain before. As mentioned by now, the iRAP pipeline is composed of several different steps, with a set of tools responsible for executing them, that are executed sequentially since every one of them depends on the previous stages due to the nature of RNA-Sequencing processes. However the changes that are going to be made to the system will allow some of the tasks to be sliced in smaller ones that can be executed in parallel, since none of the sub-tasks depend on the others which is the case of, for example, the Quality Control phase. In the mentioned stage, a quality filtering is performed over every biological read that have been collected and which are intended to be used in the RNA-Sequencing processes. Although this step is performed as a whole, representing this way only one task, it can be divided into multiple sub-tasks, where each of the reads being quality filtered would represent a task. Since the filtering process of each read does not depend on the filtering process of the other reads, the tasks splicing in the Quality Filtering stage would allow the conversion of a single task into multiple sub-tasks that could be parallelized. Being so, the tasks splicing, that is not possible only in the mentioned stage of the pipeline, coupled with the possibility of distributing the computation and the implementation of the scheduling algorithm, would allow each sub-task to be easily mapped, by the scheduler to a different node in the cluster, at the same time, transforming the execution of that stage into a parallel execution being performed in multiple computers.

Aiming at better understanding the theoretical improvements, concerning RNA-Sequencing processes' execution time, that can be achieved with the implementation of the set of actions mentioned above, the previously mentioned task splicing coupled with the scheduler implementation and the distributed computation, some of the already presented tests executed to iRAP's pipeline should be taken into consideration.

Table 3.3: A Fragment of the executed tests

Id	Mapper	Quantifier	Differential Expression
4	Tophat2	Cufflinks2	Cuffdiff2
9	GEM	NURD	DEseq2
11	GSNAP	HTSeq2	DEseq2
14	Tophat1	Cufflinks1	Cuffdiff1
15	GEM	Cufflinks1	Cuffdiff1
16	Bowtie2	Cufflinks1	Cuffdiff1
21	GSNAP	HTSeq1	EdgeR

In the previous table 3.3 a subset of all the executed tests, that have been already shown in the previous subsection for monitoring the execution times for each stage of the pipeline using different tools for the purpose, are now presented here so that they can be used to exemplify the possible gains that may theoretically be achieved. Despite the fact that the configuration of the pipeline is being presented in the table above, it is only being shown for making it easier to identify

them on the original tests' table 3.1, since for this matter the pipeline configuration doesn't matter as only the possible gains for the Quality Filtering stage is being analyzed, for exemplification purposes.

Table 3.4: Possible improvements in Filtering stage

Id	Filtering	2 Nodes	4 Nodes	6 Nodes
4	5min 30sec	2min 45sec	1min 50sec	55sec
9	5min 36sec	2min 48sec	1min 52sec	56 sec
11	6min 34sec	3min 18sec	2min 12sec	1min 6sec
14	5min 30sec	2min 45sec	1min 50sec	55sec
15	5min 30sec	2min 45sec	1min 50sec	55sec
16	5min 30sec	2min 45sec	1min 50sec	55sec
21	7min 0sec	3min 30sec	2min 20sec	1min 10sec

In the table 3.4 above it is shown the actual execution time needed to perform the Filtering stage, the one being given as example, which have already been presented in the last subsection. However the main goal that is intended to be achieved is to compare it to the theoretical execution time of the same stage of the pipeline when in the already mentioned conditions, considering three different cluster sizes two, four and six nodes. It is even of great importance to mention that, in the executed tests only six biological reads were available, so in the Filtering stage six different reads were being analyzed. Being so for the estimation of the execution time, in the three different scenarios, the splicing of previously existing task (the whole Filtering stage with the analysis of the six different reads) have been considered to produce six different parallel sub-tasks one for each analysis being performed to each read.

As mentioned, the estimation presented have been calculated considering the six reads that needed to be quality filtered in the stage currently at focus, for three different homogeneous cluster configurations (two, four and six nodes). In order to obtain these projected execution times the first step was to, for each of the executed tests, calculate the time needed for the filtering process per read to finish, which given the time needed for the Filtering stage and the number of reads filtered became easy. With this information in mind, without forgetting that the spliced tasks can be executed in parallel it was only necessary to evaluate the ratio between the number of sub-tasks, the filtering process being run on the six reads, and the number of nodes in the cluster. This way it would be easy to understand how many reads each node would need to filter and multiply the maximum of it by the already computed time needed for a read to be filtered.

At a first glance the obtained results may not seem so impressive when picturing the whole pipeline execution, since the filtering stage, in every one of the executed tests didn't come up as being one of the most time consuming stages. Although, as previously mentioned not only the Quality Filtering stage can be split, almost every stage that compose the RNA-Sequencing pipeline can be object of the same process with the goal of transforming a single task into multiple smaller parallel actions. Being so, if this process is performed for every or almost every phase of

the pipeline the positive effect onto the pipeline's execution time would be much more significant which makes it easily understandable the fact that, due to the speed up that theoretically can be achieved, the implementation of a scheduler is, as well, a valid solution.

Furthermore, the possible gains that can be achieved through the whole processes mentioned above, coupled with the possibility of executing more than one pipelines at the same time taking advantage of the scheduling algorithm, fact that can even boost even more the execution of RNA-Sequencing pipelines. Focusing on the stated during this subsection the implementation and use of the specified system seems to represent a strong possible solution, as far as performance enhancement and *makespan* reduction are concerned.

Chapter 4

Implementation

In the previous chapter [3] the problem that is meant to be worked out along this thesis project have, after a detailed analysis, been in-depth specified as well as some possible solutions that could lead to the achievement of this dissertation main goal, which is to enhance RNA-Sequencing process' performance though the improvement of iRAP pipeline's one, which is one of the available tools for executing the RNA-Seq process.

Due to the context and time limitations associated with this dissertation, the two pointed out approaches couldn't be both implemented in the existing time for the completion of the dissertation. Being so, both alternatives needed to be carefully scrutinized aiming at understanding if they were both doable and which one of them could possibly provide better results as far as performance gains are concerned.

Right from the beginning the favoritism seemed to recall onto the use of parallelism in order to improve iRAP's performance and consecutively RNA-Sequencing process by taking the most out of the existing software and hardware. The referred favoritism is explained by the differences of the two different possible approaches that can be followed with the hope of achieving this thesis main goal. First of all the use of parallelization techniques in any of the critical stages pointed out during the analyze made to the iRAP pipeline, which have been *Cufflinks* and *Cuffdiff* tools in the quantification and differential expression stages respectively, would directly improve the performance of these tools in every execution as well as in every pipeline that contains them in its configuration, which obviously represent a direct improvement of the iRAP pipeline's performance reducing this way the execution time. For the previously mentioned gains in performance to be achieved by parallelizing any of the tools, nothing needs to be improved or even changed for the performance gains to take place and to be noticed, regarding execution's time reduction, besides the parallelization of the tools that represent the critical stages of the execution.

On the other hand, when focusing on the second possibility previously established, which consists on the implementation and usage of a scheduling algorithm in order to achieve the best

execution schema, having in mind the tasks execution costs and dependencies, as a mean of speeding up the execution of the RNA-Sequencing process, the simple implementation of a scheduler does not provide any direct improvement in the pipeline's performance. Being so to take good advantage of the scheduling algorithm that would be implemented, several tasks needed to be performed besides the simple development of the scheduler. As referred already in the past chapter, the RNA-Sequencing analysis that need to be done should be executed on a cluster of processors or computers rather than on a single machine, and preferably several analysis should be done at the same time and the tasks sliced into smaller parallel ones, trying to get the most benefits out of the scheduling algorithm and the extra computational power available. It becomes obvious that a control module need as well, besides the scheduler, to be implemented with the aim of executing the tasks in the right order on the scheduled node of the cluster, taking advantage of the capabilities of a cluster on the distributed computation topic. Besides the need of implementing the scheduling algorithm and developing the control module, there is still a crucial step that needs to be performed so that the system works as whole aiming at reducing the execution times while improving its performance. Since the whole iRAP pipeline is executed as whole process encapsulating all the individual steps contained in it and which are necessary so that the RNA-Seq process is completed, it would not be possible to schedule the individual task that compose an execution, this way the execution would need to be split so that the individual steps contained in it can be properly scheduled.

From what have been mentioned regarding both alternatives, it is easily understood the reason why the simple parallelization of *Cufflinks* and *Cuffdiff* is the most tempting solution, since it would provide direct gains in performance taking advantage of the existing software and hardware without the need for a cluster and without the demand of implementing and developing extra modules to work along with the chosen scheduler.

Despite the first solution that came to light turned out to be the most tempting one, its feasibility needed to be analyzed in order to make sure that the so desired performance gains could actually be achieved with its implementation. Since the parallelization of the tools represents the core of the first solution proposed, in order to understand the viability of the proposed resolution it was of major importance to get to know, at an early stage, if the tools that were meant to be parallelized weren't already in any way parallelized. Being so, some tests have been executed, for both *Cufflinks* and *Cuffdiff*, while the CPU usage percentage was being monitored and registered to a log file, in order to understand if each tool was already taking, or not, advantage of the multicore architecture, that nowadays is found in almost every single central processing unit (CPU).

Table 4.1: Cpu usage on Cufflinks for quantification

Id	Tool	CPU Average Usage	Average
1	Cufflinks	26.10%	
2	Cufflinks	31.87%	
3	Cufflinks	27.23%	
			28.40%

Table 4.2: Cpu usage on Cuffdiff for differential expression

Id	Tool	CPU Average Usage	Average
1	Cuffdiff	62.06%	
2	Cuffdiff	65.52%	
3	Cuffdiff	60.00%	
			62.53%

For this matter three different tests have been executed to each of the tools, recording the CPU usage, obtaining the results shown in the previous tables 4.1 and 4.2. The values that can be observed in both tables, represent the average percentage of CPU usage for each one of the tools during the executed tests, and have been acquired resorting to log files that Unix systems already create containing the desired information. As mentioned, for each one of the tools three individual executions have been performed, this methodology have been used instead of doing just one test, in order to outwit any errors that could come up from any random tasks that the Operating System could be executing at the moment of the test execution, this way the average results of the three executions should provide a much more reliable outcome.

For both of the tools it is easily understood that, when comparing the different executions accomplished for each tool, the fluctuation on the presented values for the CPU average usage is not very noticeable which indicates that all the tests have been executed in similar conditions without any noteworthy external factor, as for example the Operating System running a punctual expensive task in background, which is an indicator that attaches some reliability to the obtained values.

In order to fully understand the meaning of the obtained values for the average CPU usage for each of the tools, it is important to gather some information about the machine or computer in which the tests have been executed, specially about its central processing unit's specifications. This is of major importance in favor of getting to know the percentage that a core represents in the whole CPU, which is crucial to understand whether or not a tool is already or not parallelized. Regarding the node at which the tests have been executed in, the CPU present in it is a core i7-3630QM, which is composed of 8 logical cores. Being so each core that makes part of the core i7 being analyzed represents 12.5% of the whole CPU.

From the tables 4.1 and 4.2, the average in both of them its the most important value since it represents the average between the results obtained from the three tests executed for each of the

tools. In *Cufflinks*, which is a tool used at the quantification stage, the average CPU usage among the three executed tests was 28.40% which may seem to be a low value when compared to the 62.53% achieved by the *Cuffdiff* tool, but however shows, being a higher value than the 12.5% that each core represents, that *Cufflinks* is parallelized. Being so it is easily understood that *Cuffdiff* is as well parallelized taking, this way, advantage of multicore architecture. Even though if further analyzed both tools average CPU usage could, maybe, be increased, since some effort have already been put through, by the developers, on the process of parallelizing both tools the possible gains wouldn't be so considerable as initially expected or even absent. Under this circumstances, the solution to explore during this thesis project is going to be second one presented, the enhancement of the pipeline's performance trough the implementation and use of a scheduler as explained before, since it is the one which seem to possibly provide the most noticeable improvements.

4.1 iRAP pipeline preparation

As mentioned earlier, the iRAP pipeline is composed of several sequential stages, however all these steps are enclosed in a *black box* during the pipeline's execution, causing the whole pipeline to look like a single individual task. Being so, it would be impossible to schedule the different tasks that compose the whole iRAP execution, this way the first crucial effort to make during this thesis project, would need to be the separation and isolation of each of the individual tasks that make part of the RNA-Sequencing process allowing them to be later scheduled using the scheduling algorithm that later is going to be implemented.

The only approach that could be taken in order to successfully achieve the goal of isolating the tasks that belong to the iRAP pipeline's execution was to perform an in-depth analysis in order to understand which tasks, and how are every one of them, being executed. Despite the already known tasks that are executed, there are still other actions taking place during the pipeline's execution that are not clearly mentioned as stages of the pipeline, for example tools that are being executed in the between of two stages with the main goal of converting the outputs of the first one so that they can be successfully used as input by the next one. Having this in mind, this actions, which existence could only be discovered through the analysis performed on iRAP, must as well be considered as tasks that need also to be scheduled using the yet to be chosen scheduling algorithm.

Initially the pipeline stages have been isolated, allowing the scheduling algorithm to be able to map their execution to the best time slot and cluster node, through the development of shell scripts with the main ambition of being responsible for successfully executing the desired tasks without the need for the user or the control module to directly execute them, providing this way an extra layer of abstraction. Despite the fact that, after the analysis performed on iRAP, the understanding of how the tools were being invoked and executed would be enough for directly execute these same tasks, this extra layer of abstraction provided by the implemented shell scripts ease the process of invoking the desired tool by automating as much as possible the process of fetching the needed input files for the execution while performing some background checks on

them. In certain situations, in the case of for example two sequential actions occurring in the pipeline that cannot be executed in parallel, in order to keep a certain degree of abstraction the execution of these two action is performed inside the same shell script that in this case performs a *black box*. As mentioned the RNA-Sequencing processes being scheduled and executed could be directly scheduled and invoked, however the developed scripts aim at easing this process, since this way only the scripts developed needed to be scheduled and invoked at the right time and cluster node.

However, even with the performed tasks' isolation associated with the implementation of the scheduling algorithm, no gains would be obtained when executing a single pipeline, since due to the natural execution schema of the iRAP pipeline, every stage depends on the previous one to execute, this way even with the possibility of taking advantage of distributed computation on a cluster, there would be no actions or stages that would be able to be executed in parallel, unless more than one pipeline would be executed at the same time, meaning that for a single pipeline a lot of computational power was not going to be used and the pipeline would take the same time as it did before in only a single machine, to execute. Being so, since at each stage of the pipeline some operations are performed over different inputs, which don't depend on each other, a great effort have been put into trying alter the developed scripts so that the main stages of execution could be sliced into multiple tasks allowing whenever possible the tasks to be executed in parallel, at different nodes of the cluster.

Table 4.3: Developed scripts

Stages	Tools	Yes / No
QC	FastQC	Yes
Alignment	bwa1	No
	bwa2	No
	bowtie1	Yes
	bowtie2	Yes
	gem	No
	gsnap	No
	mapsplice	No
	osa	No
	star	No
	tophat1	Yes
	tophat2	Yes
Quantification	cufflinks1_nd	Yes
	cufflinks1	Yes
	cufflinks2_nd	Yes
	cufflinks2	Yes
	flux_cap	No
	htseq1	Yes
	htseq2	Yes
	kallisto	No
	nurd	No
	rsem	No
	stringtie	No
DE	cuffdiff1	Yes
	cuffdiff2	Yes
	deseq	No
	deseq2	Yes
	edger	No
	voom	No

Due to the time limitations associated with this thesis project, and due to the huge amount of tools that are supported by iRAP, for some of them to be executed they still must be invoked directly since a shell script to encapsulate its invocation wasn't developed. Nevertheless this fact doesn't affect at all the usage of these tools, in the cluster context, since they can always be invoked directly, furthermore, as presented in the table 4.2 above, scripts are available for a considerable amount of tools that is easily enough for the execution of the tests at the validation stage, at the end of this thesis project that will reveal the true benefits of the solution implemented.

During this first stage of implementation of the chosen solution for the problem presented in the previous chapter [3], the iRAP pipeline which execution is intended to be scheduled using the scheduling algorithm needed to be prepared so that, scheduling it could become a feasible process. Due to the fact that the whole pipeline execution represented a *black box* enclosing every single action and tool invocation that was being performed during the whole pipeline execution,

iRAP execution's stages needed to be isolated in favor of being possible to schedule them. For this matter some shell scripts have been developed to execute the mentioned stages of the pipeline, always with great a amount of attention being given to task splitting aiming at maximizing the amount of sub-tasks that can be executed in parallel.

4.2 Scheduling algorithm

4.2.1 Choosing the most suitable algorithm

During the specification of the possible solutions that, if implemented, may be able to solve the stated problem, which resolution represents the main goal of this thesis project, the hypothetical gains and improvements that could be achieved with the implementation and use of a scheduling algorithm have been analyzed, as well as some modifications that can be applied to the previously existing assets, which coupled with the use of the scheduler should provide considerable gains, as far reducing the overall execution time is concerned.

Despite the fact that the use of a scheduler seems to be, at least from a theoretical point of view, a valid option when improving the RNA-Sequencing processes is the main concern, the choice of the scheduler to be used or implemented plays an unquestionable important role in the possible gains that can be achieved with it. For starters a very computationally complex scheduler can become a bottleneck in the system nullifying the achievable gains. It is even possible that some schedulers don't offer as much features or customization options that could be important in order for the conceivable amount of gains to be, as much as possible, enlarged.

Some important choices needed to be made upon the process of choosing the scheduling algorithm that was implemented during this thesis project. As already mentioned, the scheduler is going to be associated with the possibility of distributing the whole iRAP's computation on a cluster, being so, the first crucial decision to be taken relies on the demand for the scheduler to be able to schedule the tasks to heterogeneous clusters.

As defined by Arabnejad et al. [AB14] an heterogeneous system can be defined as a set or an aggregation of different resources, which can be local or geographically distributed, serving the main purpose of executing computationally intensive applications.

The core difference between a heterogeneous and a homogeneous cluster is related to the used nodes, in this case the computers that compose the cluster and which are going to be used to execute the RNA-Sequencing tasks. While, as stated, in a heterogeneous cluster each node can represent different resources such as different computational power for example, in an homogeneous system every node must be equal to the others, adding exactly the same value to the whole system. Since there is no guarantee that RNA-Sequencing processes are going to be executed in an homogeneous cluster, and since a scheduler that is able to map the tasks to the most appropriate time fraction and node, in an heterogeneous cluster, is as well going to be capable of doing so in

an homogeneous one, a great amount of attention needed to be given to this matter in order to find a scheduler that supports heterogeneous systems.

Despite the fact that this way the set of clusters at which the RNA-Sequencing processes can be executed is not going to be limited by their configuration, a scheduling algorithm suited to heterogeneous systems is certainly more complex which may create some issues as far as improving the iRAP pipeline's performance is concerned. As mentioned by Arabnejad et al. [AB14] the higher complexity associated with the process of scheduling tasks into heterogeneous systems, when in comparison to an homogeneous one, is related to the different execution rates among processors and possibly different communication rates among different processors.

Besides the fact that the stated above represent a great influence in the scheduler complexity, there is still another major choice with great impact on the complexity of the scheduler as well, that should be taken. Regarding the information needed and used by the scheduling algorithm to perform its job, there are two type of schedulers, the static and the dynamic ones.

Regarding the static category of schedulers, all the information needed about the tasks to schedule, such as its execution and communication costs, as well as its relationships with other tasks must be known beforehand. On the other hand, for the dynamic scheduling algorithms, such data is not needed to be available at the begging of the execution, and all the decisions are made at the runtime. Being so, static scheduling is a compile-time process, whereas dynamic scheduling represents a runtime scheduling.

It is easily understood that dynamic scheduling algorithms are much more complex than the static ones, requiring and using much more resources than what would be needed to execute a static scheduler. Furthermore, a dynamic scheduler would need to be constantly running, since it computes and provides the schedule in runtime, at the same time at the pipeline would, consuming resources that could be used to execute some RNA-Sequencing tasks. Being so and since every information about the pipeline's execution can be be known beforehand to the execution the choice falls onto a static scheduler which allows a less complex scheduling algorithm to be used, saving this way computational resources.

As mentioned the complexity of the scheduler, as far as resources and time are concerned, is of great importance since its imperative that the scheduler don't represent in any way a bottleneck in the system, as otherwise the whole process of implementing it would be useless. Being so, there was the need to find some scheduling algorithm with the previously mentioned attributes and capable of providing an acceptable scheduling while not jeopardizing its complexity and consequently the possible gains of the whole system.

In the search for low-complexity scheduling algorithm, some already conducted studies have been analyzed, aiming at encountering a suitable scheduling heuristic or algorithm. Canon et al. [CJB07] conducted a study with the main purpose of comparing the complexity and the quality of the output schedule of twenty distinct scheduling heuristics in the pursuit for a low-complexity but at the same time robust one. In the performed analysis, which have been already mentioned,

HEFT algorithm, which is a list-based heuristic and supports heterogeneous clusters, have been declared the best one as far as average *makespan* and robustness is concerned.

Confirming what Canon et al. concluded from the conducted study [CJB07] back in 2007, also Arabnejad et al. [AB14] stated, more recently, that list scheduling heuristics are able to produce the most efficient schedules, as far as the output schedule quality is concerned, without compromising the *makespan* and with a complexity that is usually quadratic in relation to the number of tasks.

Even though, HEFT was believed, bt Canon et al. [CJB07], to be the best heuristic, Arabnejad et al. [AB14] presented a new list scheduling algorithm that is meant to work, as well, for connected heterogeneous processors, allowing the tasks to be properly scheduled to the nodes of any heterogeneous cluster. The newly developed algorithm called Predict Earliest Finish Time(PEFT) is meant to outperform HEFT as far as the overall time execution and efficiency are concerned, while sharing the same complexity with it.

The better results that are achieved by PEFT when in comparison with the previously mentioned HEFT algorithm have to do with the fact that, in the new one, a lookahead feature managed to be added without increasing the time complexity of the algorithm when comparing to HEFT, which is the real novelty that was added, meaning that it would take the same time and resources to schedule the tasks while providing better and more robust results. Being so PEFT is able to outperform all the list-based scheduling heuristics with quadratic time complexity, since besides itself no other algorithm evaluates what is ahead, as far as execution flow is concerned, assessing only the current task, demonstrating a greedy behaviour, which in some cases leads to poor scheduling decisions.

Arabnejad et al. [AB14] not only state that the developed list scheduling algorithm (PEFT) improves the scheduling provided by every algorithm with quadratic complexity, such as HEFT, but also that PEFT is the first heuristic able to outperform HEFT, the one which was believed to be the best scheduling algorithm, while being able to maintain the same time complexity.

Through the whole in-depth analysis that have been fulfilled aiming at bringing light upon the existing alternatives among the available scheduling algorithms, which would respect every need previously mentioned, a considerable importance have been given to the schedulers' performance and complexity since, as explained, it is of major importance that the chosen algorithm is able to perform fairly well as far as scheduling is concerned while not requiring too much time and resources to be executed. Being so, the Predict Earliest Finish Time (PEFT), seemed to be and indisputable choice for the current scenario. This way, PEFT algorithm was the one implemented aiming at scheduling the tasks that make part of the RNA-Sequencing process onto a cluster constituted of several nodes, that may or not, be heterogeneous.

4.2.2 PEFT Implementation

PEFT's implementation ended up to taking a considerable amount of the whole time available for the conclusion of this thesis project.

Algorithm 1 PEFT Pseudocode Algorithm

Require: Compute OCT table
Require: Compute $Rank_{oct}$ for all tasks
Require: Create Empty list *ready-list* and put n_{entry} as initial task
  while *ready-list* is NOT Empty **do**
    $n_i \leftarrow$ the task with the highest $Rank_{oct}$ from the *ready-list*
    for all processor p_j in processor-set P **do**
      Compute EFT(n_i,p_j) value using insertion-based scheduling policy
      $O_{EFT}(n_i,p_j)$ = EFT(n_i,p_j) + OCT(n_i,p_j)
    end for
    Assign task n_i to the processor p_j that minimize O_{EFT} of task n_i
    Update *ready-list*
  end while

The development of the scheduling algorithm have been split into two different stages. During the first stage, which turned out to be the most time consuming one, the implementation of the chosen scheduler took place, following its detailed algorithm, presented above, which have been taken from the original specification [AB14]. The scheduler have been implemented using the Java programming language, in order to take advantage of the *Write once, run anywhere* concept, mainly for the sake of obtaining a scheduler that can be executed in all platforms, that supports Java, without having the need of recompiling it. During the development of the scheduler the algorithm have been strictly followed in order, not only to obtain the expected results, as far as schedule outputs are concerned, but also to maintain the so crucial algorithm's complexity, which in PEFT represents one of its most valuable assets as it is coupled with the previously explained lookahead.

The implemented scheduling algorithm works upon a DAG (directed acyclic graph), in order to be able to produce the desired output representing the tasks execution's schedule. Being so, a built-in module have been developed and added to the scheduler, as well during the first stage of the implementation, that is in charge of receiving the set of tasks to schedule and based on their dependencies create a valid DAG that can be processed by the scheduling algorithm. The mentioned DAG that is created before the execution of the scheduling algorithm itself, is created based on a .XML file that is passed as an input to the scheduler, which execution starts from the responsible module for the parsing the mentioned file, which coupled to it.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<xml>
  <schedulingoutput>output/output.xml</schedulingoutput>
  <sharedDir>sharedDir</sharedDir>
  <serverUsername>root</serverUsername>
  <sshPort>22</sshPort>

  <processors>
    <processor>
      <name>Node2</name>
      <ipaddress>172.30.3.67</ipaddress>
      <username>johndoe</username>
    </processor>
    <processor>
      <name>Node1</name>
      <ipaddress>172.30.3.52</ipaddress>
      <username>janedoe</username>
    </processor>
  </processors>

  <dag>
    <tasks>
      <task>
          <name>T1</name>
          <output>../../T1</output>
          <script>t1script.sh</script>
          <inputs>
            <input>input1</input>
            <input>input2</input>
            <input>input3</input>
          </inputs>
          <dependecies>
          </dependecies>
          <costs>
            <cost>
              <node>Node1</node>
              <value>52</value>
            </cost>
            <cost>
              <node>Node1</node>
              <value>81</value>
            </cost>
          </costs>
        </task>
     </tasks>
  </dag>
</xml>
```

Listing 4.1: Example input .XML

The .XML file presented in the listing 4.1 aims at demonstrating the structure of the input file that is expected to be received by the scheduler. Despite not being crucial for the scheduling algorithm to perform the tasks schedule with complete success, some information have been added to the .XML file exhibited, since some of that information is of major importance for the control module, that have later been implemented, in order to be able to correctly execute the set of tasks according to the scheduler output. As the .XML file shown is only meant to demonstrate the file's structure that is accepted by the scheduler, only a single task is this way displayed. Regarding the demanded information for the successful schedule of the whole set of tasks, the available nodes, belonging to the cluster available for the execution of the RNA-Sequencing processes, coupled with the information about the costs' forecast for executing each tasks in each node is critical for the scheduling process. Together with the mentioned data, the dependencies between the actions to be scheduled are, as well, crucial information that must be available in the .XML file passed as input to the scheduler.

At the end of the first stage of the scheduler development, it was already possible to successfully schedule a complete set of tasks, representing a pipeline execution. Being so the scheduling algorithm was already fully implemented as well as the .XML parser module that at an initial stage of the scheduler is executed in order to parse the input file while creating the directed acyclic graph (DAG) that is being used later by the scheduling algorithm. Despite the fact that the implemented scheduler was already fully capable of performing its duty, there was still a major improvement that could be implemented focusing on boosting the performance gains and *makespan* reduction that the system under implementation could provide.

Being so, in the second stage of the scheduler implementation, in order for the new desired improvements to be able to be integrated to the developed scheduler, the input .XML file that was, at that time, being accepted as input needed to be reconstructed. The new established format is then presented in the listing 4.2 aiming at mainly pointing out the differences that took place from the first stage of implementation to the second one. This way some detailed information about the tasks themselves, that have been shown in the previous .XML, have been suppressed, in the one presented now, as there were no changes affecting their structure. The only difference that can be perceived when analyzing both files relies on the fact that, at the second stage of the implementation, the .XML input files may have contained in it multiple "dag" tags. Despite the fact that it seems quite a simple modification that have been introduced during the last stage of implementation, the incorporated innovation, forced the parser module itself and the scheduling algorithm to be changed to go along with the changes being made to the input. The mentioned modifications allowed the scheduler developed to be able to schedule multiple sets of tasks, in a single execution, which means that multiple RNA-Sequencing pipelines are going to be mapped to the available resources in the cluster, at the same time, with a single execution of the scheduling algorithm.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<xml>
  <schedulingoutput>output/output.xml</schedulingoutput>
  <sharedDir>sharedDir</sharedDir>
  <serverUsername>root</serverUsername>
  <sshPort>22</sshPort>

  <processors>
    <processor>
      <name>Node2</name>
      <ipaddress>172.30.3.67</ipaddress>
      <username>johndoe</username>
    </processor>
    <processor>
      <name>Node1</name>
      <ipaddress>172.30.3.52</ipaddress>
      <username>janedoe</username>
    </processor>
  </processors>

  <dag name="dag1">
    <tasks>
      <task>
        ...
        </task>
        <task>
        ...
        </task>
     </tasks>
  </dag>
  <dag name="dag2">
    <tasks>
      <task>
        ...
        </task>
        <task>
        ...
        </task>
     </tasks>
  </dag>
  <dag name="dag3">
    <tasks>
      <task>
        ...
        </task>
        <task>
        ...
        </task>
     </tasks>
  </dag>
</xml>
```

Listing 4.2: Example input .XML on second stage

The development of the scheduler represents the second stage of the implementation of the proposed solution, for the problem previously specified in this thesis project. Despite the fact that the solution being implemented depends directly on every one of its stages, the implementation of the scheduling algorithm, performed during the second one, represents the core of the designed solution. During this stage, besides the chosen scheduling algorithm some features have been thought of, in order to be implemented and accepted by the scheduler itself. The scheduler is composed of two different modules, one corresponding to the scheduling algorithm and a different one aiming, not only, at parsing all the information, received as input, regarding the cluster's nodes and pipeline tasks to be executed which result is later used by the scheduling one, but also at providing an output in a previously delineated format, that can be used by the control module, which is the final step to be consummated, as far as the solution being implemented is concerned. Even though the scheduler have been implemented aiming at being an asset for the solution being implemented, with the main goal introducing improvements as far as *makespan* reduction and performance enhancement are concerned, its implementation have being performed as generic as possible allowing the implemented scheduler to be used to schedule any set or sets of tasks without requiring that they belong to iRAP or even RNA-Sequencing analysis.

4.3 Control Module

The control module represents to the solution implemented as a mean of mitigating the main problem presented earlier in this dissertation, its third step. Despite being the last stage in the development of the chosen solution, at this stage of the implementation, all the structures necessary for the performance gains to be possibly achieved, have already been implemented in the two previous stages; However without the development of this last stage, which is necessary for the aggregation of the whole system, it would not fully work and be able to execute the RNA-Sequencing processes that are meant to be executed. This module is the one that the user is going to have contact with, since it encapsulates every single process that need to be executed for the whole system to work, this way its implementation is of major importance.

Due to the same reasons, that were previously presented above concerning the implementation of the scheduler, also the control module have been implemented using Java programming language exploiting the advantages of the *Write once, run anywhere* concept. As already mentioned, this module is going to, during its execution, carry the load, for the new system, of invoking every single action and step required from the very beginning of the execution until the end of the last pipeline's stage execution, invoking this way all the sets of tasks that are contained in the input .XML file. Being so, it is understandable with ease that for the stated above to be possible the module currently on focus must internally, enclosing it from the user, execute the scheduler, analyze its output and based on it, execute each task in the right node of the cluster with complete respect for the output of the scheduling algorithm.

In order to be executed, by the user, the control module requires that a .XML file is provided as input. Having in mind both the fact that the scheduler itself must receive, as well, an .XML file and that the scheduler is going to be internally invoked by the control module, it would be intuitive to assume that to execute the system an extra input file would need to be provided in order to later be passed as input to the scheduler. However aiming at easing all the process of executing the system, in order to execute the desired set or sets of tasks, the sub-module responsible for the parsing of the input file was developed in such a way that it would accept the same .XML file that the scheduler will need to receive, which wipes out right away the need for the user to provide two different input files to the control module. This was only possible since the information needed by the control module represents a subset, although more detailed, of the whole information required for the scheduler to be executed.

For the purpose of the control module developed to be successfully achieved its execution have been partitioned into multiple different stages. In the first stage of the system's execution the input file, which structure have already been presented in the listing 4.2, provided by the end user is parsed with the aim of obtaining every information needed about the cluster and the nodes available to execute the tasks required.

Due to the nature of the tasks to be executed for the RNA-Sequencing analysis, typically every stage depends on the output of the immediately preceding one, having this in mind and assuming that being executed in a cluster the tasks may be executed in different nodes, it is clear that some kind of file exchange need to exists otherwise the tasks that need to be executed will not be able to have access to the required files for their execution. Being so, the solution adopted was to empower the server with the role of creating and exporting a NFS (network file system) that every node of the available cluster must mount. The connection to the remote file system as well as the ssh connection, between the server node and the remaining nodes of the cluster, that is going to be used to execute the scheduled tasks, must already be configured and established upon the invocation of the control module. This way the second step, being performed after the retrieval of the nodes information through the analysis of the input .XML, is to test both the NFS and the SSH connections. In order to be possible at *runtime* to check, for every node, that the network file system exported by the server is correctly mounted and that the ssh connection to the server is fully functional a file is being created by the control module, that will be running in the server, through ssh connection on the mounted directory for each of the nodes; Lately every file existence is being checked which will reveal if all the connections are well-established.

If every connection happens to be tested out and working in the previous stage, the control module is going to carry on its execution to the next stage, the third one. This stage represents one of the core stages of the system, in which the scheduler is going to be invoked and its output is going to be collected in order for the tasks to be later executed according to the output gathered. The scheduler's output is being computed and then exported to a .XML file that down the road will be parsed by a built-in parsing module contained in the control module.

```
<tasks>
  <task>
    <name>T1</name>
    <dag>dag1</dag>
    <output>../../T1</output>
    <script>t1script.sh</script>
    <startTime>0</startTime>
    <endTime>52</endTime>
    <node>node1</node>
    <inputs>
      <input>input1</input>
    </inputs>
    <successors>
      <successor>dag1.T2</successor>
    </successors>
  </task>
  <task>
    <name>T2</name>
    <dag>dag1</dag>
    <output>../../T2</output>
    <script>t2script.sh</script>
    <startTime>52</startTime>
    <endTime>94</endTime>
    <node>node2</node>
    <inputs>
      <input>../../T1</input>
    </inputs>
    <dependencies>
      <dependency>dag1.T1</dependency>
    </dependencies>
  </task>
  <task>
    <name>T1</name>
    <dag>dag2</dag>
    <output>../../dag2T1</output>
    <script>dag2t1script.sh</script>
    <startTime>0</startTime>
    <endTime>40</endTime>
    <node>mode2</node>
    <inputs>
      <input>input2</input>
    </inputs>
    <successors>
          <successor>dag2.T2</successor>
    </successors>
  </task>
  <task>
    <name>T2</name>
    <dag>dag2</dag>
    <output>../../dag2T2</output>
    <script>dag2t2script.sh</script>
    <startTime>40</startTime>
    <endTime>84</endTime>
    <node>node1</node>
    <inputs>
      <input>../../dag2T1</input>
    </inputs>
    <dependencies>
      <dependency>dag2.T2</dependency>
    </dependencies>
  </task>
</tasks>
```


Listing 4.3: Example scheduler .XML output

The previously exhibited .XML contained in the listing 4.3 represents an example of the output exported by the scheduler after the execution of the scheduling algorithm. It reveals itself with a very simple structure with only the strictly necessary information about every single tasks that is supposed to be executed and the information about where and when to execute it. Despite the fact that the scheduling algorithm supports the computation of the predicted start and finish time, this information is not going to be directly used, as far as tasks invocation is concerned in the last stage of the control module execution.

The system's last stage is in charge of executing all the tasks and actions respecting strictly the parsed output obtained from the scheduler. As mentioned above, the estimated start and finish time are not going to be used for the control module to know when to execute each task, since as it is an estimation some precedent task can end its execution earlier than predicted and computational time would be lost if the scheduled time was being followed. Being so, the scheduler's output is being parsed and for each of the available nodes in the cluster, a prioritized list is being created following the tasks execution order. Having in mind that for each of the nodes, a list is available with the tasks to be executed correctly ordered, it is easily understood that tasks are being executed as soon as the task immediately before in the list, for the same node, finishes it execution. However, despite the fact that the stated above came in handy as no computational time was being unnecessary lost, the implementation done brought some problems relating to the dependencies between the set or sets of tasks to execute, as this way some tasks could be invoked before its dependencies were finished causing unauthorized accesses to files that haven't been at the time created. Therefore some restrictions were added to the control module in its last stage of development to mitigate the previously specified problem.

Through the structure and features implemented in the control module it became possible that the whole execution of the system could be encapsulated, enclosing every step from the end user which warrants a much more user friendly system. To execute the system it is only required that the user invokes the control module with an .XML file containing every information about the nodes available and the tasks that should be executed, as presented above, since the control module will, itself, execute every check necessary, invoke and fetch the scheduler's output and at a final stage, using NFS and SSH connections execute every task, respecting the scheduling output, in the right node of the cluster.

4.4 Summary

Two possible solutions were thought of aiming at extenuate or even mitigate the performance problems associated with RNA-Sequencing processes, which represents the main problem of this thesis project, however only the most promising one, which was the second alternative (the implementation of a scheduler) have been implemented.

Having into consideration that the tasks to be scheduled are RNA-Sequencing processes, composing stages or actions being performed during an execution of iRAP pipeline, it is known that

each of them depends, as long as inputs are concerned, to the tasks and stages performed immediately before, as the output of a stage of the pipeline is needed as input for the following step of the execution. This way, the implementation and use of a scheduling algorithm by itself, in an attempt of scheduling every task composing an RNA-Seq analysis would not provide neither any performance gain nor any improvement in the execution time of the whole set of tasks. Being so, in order for a scheduler to be able to introduce some benefits in the system, some modification needed to be made to it, changing the way RNA-Sequencing processes were being executed.

The changes implemented allowed not only the developed system to be able to execute multiple RNA-Sequencing processes at a time, parallelizing multiple pipeline's executions, but also some stages of the pipeline to be split into smaller parallel tasks that would be executed in parallel on the available cluster. Taking advantage of the speedups that are expected to be obtained, in order to improve the RNA-Sequencing analysis' results regarding its quality and accuracy, the same task can be scheduled multiple times in parallel, performed with the same tools or not, and a voter scheduled after their execution to improve the quality results.

Right away, at the beginning of the implementation process, it have been noticed that every task and action performed during the pipeline, were being encapsulated in a single task which would nullify the possibility of using a scheduler. To overcome this issue a detailed analysis have been performed on iRAP aiming at isolating each of the stages of the execution, creating scripts capable of executing them, which would allow the use of a scheduling algorithm.

The choice of the scheduling algorithm to use in this system, represented an important decision to make since it could affect the whole performance of the system under development. Focusing on implementing a scheduler that could be able to provide good results, as far as the output schedule is concerned, while not harming the system's performance trying to keep the its complexity as low as possible, PEFT algorithm was the one implemented.

In the last stage of the implementation of this dissertation, a control module have been developed in order to adjoin the whole system. Its main purpose is to distribute the computation through the nodes of the cluster available, taking into attention the scheduler output. Being so, this module is not responsible for the tasks scheduling, but for executing them at the right time and at the right cluster's node. The developed module have been implemented in a way so that its the only piece of software that the user have contact with, encapsulating in a *black box* every action taking place during the system's execution; therefore it is going internally, hiding it from the user, to parse every input needed, gathering information about the cluster and tasks to be executed, before invoking the scheduler that will return the output schedule to be followed by this control module upon distributing the tasks' execution the cluster nodes.

4.4.1 Before and After implementation

The implemented solution, aiming at solving the problem that this thesis purposes to mitigate brought some changes to the previously existing system that, despite having been previously specified in-depth, are going to be shortly shown in the diagrams presented bellow.

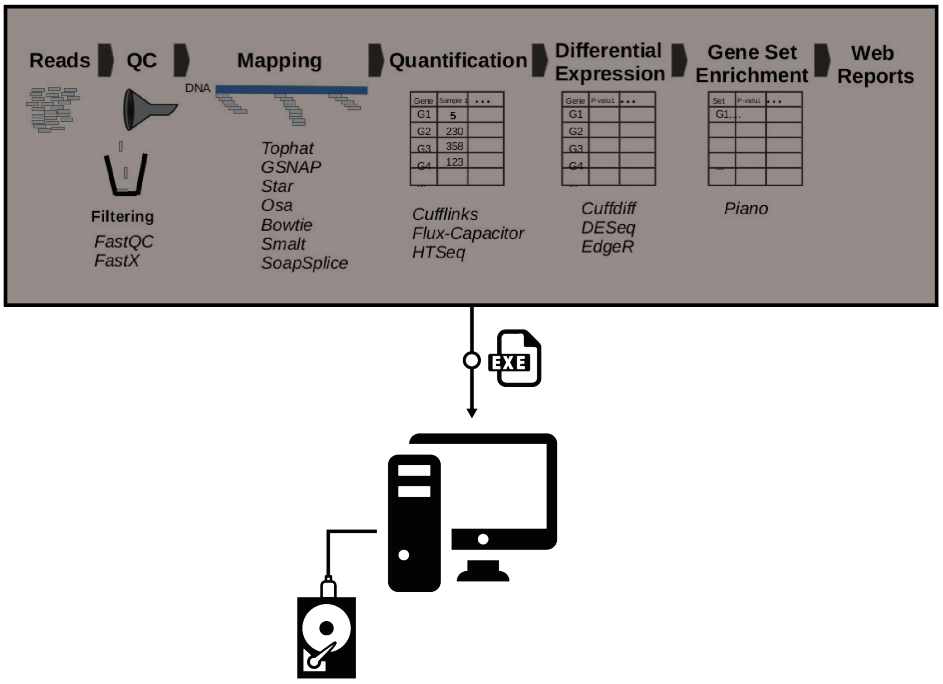

Figure 4.1: System before the implemented solution

The previous figure 4.1 represents the system as it was before the implementation of the chosen solution. As perceptible from it, the pipeline was previously a *black box* to the end user, which could only recognize and execute a task that internally would execute every required actions for the whole pipeline execution to be successfully performed. The pipeline's execution would be performed in a single machine with a local filesystem.

Regarding the solution implemented, to take advantage of the implementation of a scheduling algorithm some changes needed to be done to the existing system that can be perceived in the presented figure 4.2. For starters the single computer being used have been exchanged for a cluster, such that one of the nodes assumes the role of server executing the control module and exporting a directory of its local filesystem as a network file system, for the other nodes to mount. Speaking of which, these must be fully configured so that they are able of executing RNA-Sequencing processes. Regarding the pipeline and tasks to be executed, these are not in a *black box* anymore in order for them to be scheduled among the cluster. After the implemented solution, the server, executing the control module, receives the information about the tasks to schedule and execute

(represented as the EXE files in the diagram), and invokes the scheduler module providing it with that information. After its execution the output is going to be parsed by the control module that following the scheduler's outcome is going, through ssh connections, to execute the right tasks at the right nodes of the cluster.

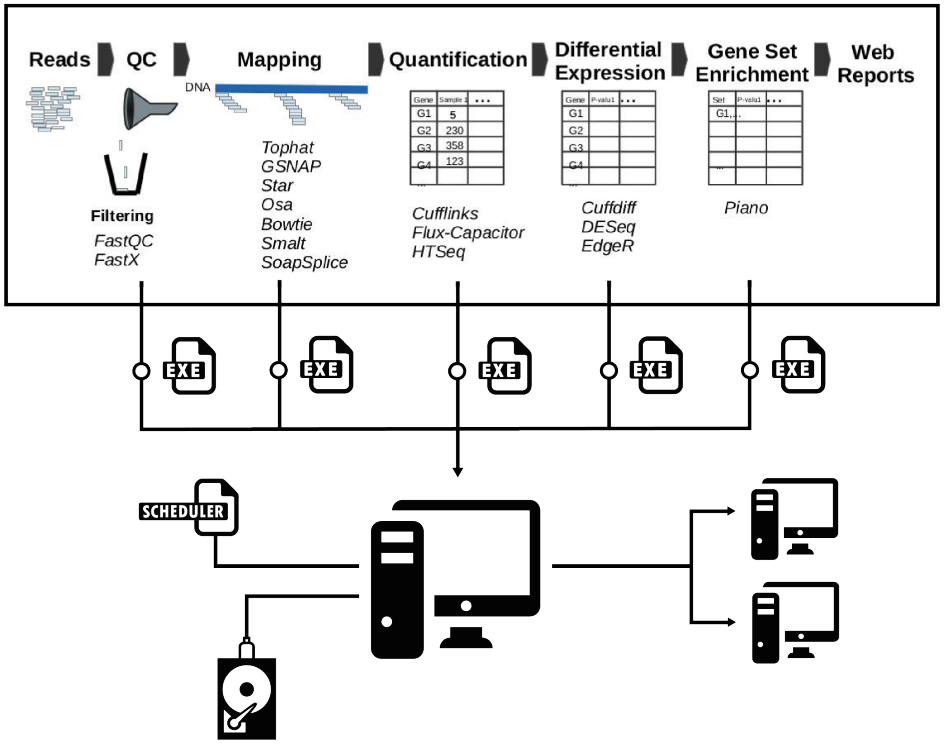

Figure 4.2: System after the implemented solution

Chapter 5

Validation

During the last chapter [4] the details about the developed solution, which have been, throughout this whole thesis project, believed to be an adequate approach to attenuate the problem at which this dissertation is focused, were presented. Although, in the current chapter, since it is of major importance to validate the implemented system, some of the executed tests for the matter are going to be presented.

As already mentioned, the developed system aims at mitigating the performance issues that are inevitably associated with the iRAP, since it is a tool that intent at implementing and executing complete RNA-Sequencing analysis. Considering that the efficiency improvement desired is meant to be rendered in a reduction of the *makespan*, the total execution time, the performed tests to the system have focused on obtaining accurate measurements of the whole execution duration. Therefore the control module developed, which main purpose is to aggregate the whole system encapsulating every action that is being executed from the beginning of its execution until the end of the last scheduled task's execution, have been adapted to be able to measure the whole execution span.

5.1 Setting up the tests environment

After the end of the implementation process, a great amount of time have been spent with the main goal of properly setting up the environment at which the tests were being executed. As mentioned in the previous chapter [4] the new system is meant to be running in a cluster of computers, that are going to be connected to the internet through ethernet cables, being so for each node of the cluster that is going to execute tasks corresponding to RNA-Sequencing processes, the iRAP pipeline as well as all its dependencies needed to be correctly installed and configured.

Besides the nodes that are actually going to execute the demanded actions, there was still the need to have a computer with the purpose of being the server node. This system's filesystem is

actually going to be the server's local filesystem that through NFS is going to be exported by the server node and mounted in each of the other cluster's constituents. This actually represent one of the newly implemented system drawback, that is later going to be further explained.

A step of major importance during the cluster's configuration, is the set up of the ssh connections between the server and the other nodes available, so that the control module, that is going be executed in the server, can be able through ssh connection to execute the right task, at the right time in the scheduled node, as specified by the scheduler that is executed in the beginning of the control module.

5.2 Setting up benchmarks

One of the most important efforts to make in order for meaningful results to be possibly achieved is to gather information about the previously existing solution, in other words, capture import data related to the iRAP pipeline's execution, which is the focus and start point of this dissertation. Being so, the first stage concerning the validation of the system implemented during this dissertation, was to set up benchmark values, so that later they can be set side by side to the values that are going to be obtained from the newly developed system.

First of all, taking into consideration that the values to collect should represent as reliably as possible the system's behavior as it was before the implementation of the new solution, the tests have been executed in a single computer of the available cluster for performing the system's validation.

In an initial stage, the execution times that were going to compose the reference results were being collected directly from the iRAP pipeline itself, taking advantage of the fact that the execution duration of each stage of the pipeline was already being accounted and registered. Despite representing an easy and most importantly a correct approach, as far as collecting benchmark values is concerned, this methodology could later, in the analysis stage, be proven to be a wrong technique which would immediately invalidate the reference values collected. This assumption came to light, since as mentioned during the implementation chapter [4] a stage, that compose the whole implementation process, of major importance is related to the isolation of each step of the pipeline execution, so that the tasks could be scheduled, this way some scripts have been implemented to ease the process of executing some of the tasks smoothing the current testing and validation phase. Therefore despite the previously existing pipeline (iRAP) and the developed scripts invoke the same tools for the same stage, those are now being invoked through the developed scripts for the purpose and not anymore being executed using the pipeline, since it needed to be split for the tasks to be able to be scheduled.

Due to the fact that, as explained, the same tools are being executed, the overall time execution should be equivalent, however some variations may occur due to for example code complexity / optimization, checks being performed in the pipeline that are not being performed in the developed

scripts, or even the decompression of some files or reads. Moreover differences can be introduced simply due to the fact that the measuring of the elapsed time is being done at different periods of the execution, in a scenario where, for example, the scripts don't consider converting the output of a tool, so that it can be accepted by the next one, in the measurements while the iRAP may do so.

The stated above is a matter of great importance due to the fact that, to test the new system the new scripts are going to be used, if considerable differences exist between the times collected based on the iRAP and, on the other hand, based on the implemented executable scripts, when comparing the results obtained for the new system with the benchmark values (assuming they would be obtained based on iRAP) the possible conclusions that could be derived from it would not be so reliable and accurate as it would be expected and desired. Aiming at understanding whether or not the constraint explained could actually occur, some tests have been performed with the main purpose of comparing the effect of using iRAP or the newly developed scripts on the execution time.

Aiming at ensuring that the results obtained from these tests are solid and unquestionable, both the measurements performed for iRAP and the created scripts have been performed in the same environment, at the same machine and for both of them 3 consecutive tests have been performed, instead of just one, to assure that no irregular event that could occur in the computer while the tests were running would influence the tests results as much, or at least without being noticed.

Table 5.1: Measurements based on iRAP

Tool	Start Time	Finish Time	Time Elapsed
FastQC	12:16:59 AM	12:22:03 AM	5min 4secs
	12:40:18 AM	12:45:13 AM	4min 55secs
	01:03:27 AM	01:08:21 AM	4min 54secs
Mean			**4min 58secs**
Bowtie2	03:34:37 AM	03:42:58 AM	8min 21secs
	04:39:40 AM	04:48:04 AM	8min 24secs
	04:16:37 AM	04:25:01 AM	8min 24secs
Mean			**8min 23secs**
Tophat2	01:31:29 AM	01:50:12 AM	18min 43secs
	01:55:19 AM	02:13:59 AM	18min 40secs
	02:19:00 AM	02:37:43 AM	18min 43secs
Mean			**18min 42secs**
HTSeq2	03:42:58 AM	03:49:28 AM	6min 30secs
	04:48:04 AM	04:57:52 AM	9min 48secs
	04:25:01 AM	04:34:49 AM	9min 48secs
Mean			**8min 42secs**
Cufflinks2	11:29:55 PM	11:42:50 PM	12mins 55secs
	12:44:30 AM	12:57:25 AM	12mins 55secs
	01:58:55 AM	02:11:51 AM	12mins 56secs
Mean			**12mins 55secs**
Cuffdiff2	09:55:23 PM	10:24:56 PM	29min 33secs
	09:05:22 PM	09:32:04 PM	26min 42secs
	08:18:38 PM	08:41:17 PM	22min 39secs
Mean			**26min 18secs**

Table 5.2: Measurements for developed scripts

Tool	Start Time	Finish Time	Time Elapsed
FastQC	07:42:48 PM	07:44:22 PM	1min 34secs
	07:44:22 PM	07:45:52 PM	1min 30secs
	07:45:52 PM	07:47:23 PM	1min 31secs
Mean			**1min 32secs**
Bowtie2	07:47:23 PM	07:51:26 PM	4mins 3secs
	07:51:26 PM	07:55:15 PM	3mins 49secs
	07:55:16 PM	07:59:07 PM	3mins 51secs
Mean			**3mins 54secs**
Tophat2	07:59:07 PM	08:15:51 PM	16mins 44secs
	08:15:52 PM	08:32:34 PM	16mins 42secs
	08:32:34 PM	08:49:15 PM	16mins 41secs
Mean			**16mins 42secs**
HTSeq2	08:49:15 PM	08:55:13 PM	5mins 58secs
	08:55:13 PM	09:01:13 PM	6mins 0secs
	09:01:13 PM	09:07:11 PM	5mins 58secs
Mean			**5mins 59secs**
Cufflinks2	09:07:11 PM	09:30:07 PM	22mins 56secs
	09:30:07 PM	09:53:09 PM	23mins 2secs
	09:53:09 PM	10:16:13 PM	23mins 4secs
Mean			**23mins 1sec**
Cuffdiff2	10:16:13 PM	10:31:27 PM	15mins 14secs
	10:31:27 PM	10:46:39 PM	15mins 12secs
	10:46:39 PM	11:01:53 PM	15mins 14secs
Mean			**15mins 13secs**

In the two previously presented tables 5.1 and 5.2 the already mentioned tests, which intent at comparing the execution times based in iRAP and using the newly developed scripts aiming at invoking the tools that can be executed at each stage of the pipeline, are presented. As to some extent was already expected, for the majority of the tested tools, the results obtained for the two different concepts vary quite considerably, which, like explained before, means that opting for using the ones based on the iRAP pipeline will not provide so precise and unambiguous conclusions as it would be desirable.

The following conditioning factors may be in the root of this disparity regarding the obtained elapsed times:

Differences in code complexity / optimization.

Checks being performed in the iRAP pipeline that may not be performed in the developed scripts, or vice versa.

Some files or reads being decompressed in one of the methodologies and not in the other one.

Measuring the elapsed time being done at different periods of the execution, one of the methodologies may consider the conversion of the inputs or outputs while the other doesn't.

The tools being executed at each stage of the pipeline support numerous types of configurations due to specific parameters that may be given at the time of its invocation, and related to the lack of in-depth knowledge among the RNA-Sequencing processes, in the implemented scripts, the tools are being executed with the default settings, which may not happen in iRAP.

Having into consideration that, as shown above, the adoption of the obtained results, based on iRAP, as benchmark values would not be ideal for later comparing it to the ones to be gathered for the newly fully developed system, it is easily understandable the the best pick as far as reference values are concerned, is to use the ones obtained through the execution of the scripts, since they can be later directly compared. Despite the fact that this approach does not represent the most obvious method, when the acquisition of benchmark values is the goal to achieve, after the executed tests it revealed itself to be the methodology more capable of providing a trustworthy and indisputable comparison with the results that hereafter are going to be collected from the execution of the new system. Moreover choosing the stated results as the reference ones have no negative influence at all as far as measuring whether or not the implemented system enables the achievement of some performance gains reducing the overall *makespan*.

5.3 Testing the developed system

After the completion of the first procedure that have been performed regarding the validation of the implemented solution, that aimed at setting up reference values, the main phase, which consists of gathering the execution times for the newly developed system among the set up environment for the purpose, of this validation process needed to be performed.

At this stage, the *makespan* for some of the available tools for each of the pipeline stages, was this way registered so that they can be compared with the benchmark values previously obtained. Which will reveal if the solution developed during this thesis project, improved, or not, the performance of the RNA-Sequencing processes by reducing its execution time. During the testing process for each of the tools three consecutive executions are going to be performed aiming at reducing possible noise being introduced, for example, for a one time event occurring at the operating system level such as the invocation of an unusual process by the OS. Although for the *Cuffdiff* tool only a single execution is going to be completed accounting the fact that this tasks cannot be spliced into smaller ones, being so no benefits could be extracted from its execution among the new developed system causing the comparison of its results to be irrelevant for the measuring of the system's performance gains.

The cluster available for testing purposes is composed of 3 different computers, however since one of them is going to be used as a server only two of them were used to execute actual RNA-Sequencing processes, while the other one, as a server, is the core of the developed system, since its the one responsible for executing the control module that coupled with the scheduler, will, after gathering its output execute the tasks in the other nodes through the ssh connections established with them.

5.3.1 1 Server, 1 Node

Despite the fact that the implemented solution is intended to be executed getting the most out of the cluster available and the distributed computation ensured by the control module, the first test performed only used a single computer to execute the RNA-Seq processes, in other words, only one node was being used besides the server. At a first glance it would be predictable that with only a single computer of the cluster executing the RNA-Sequencing processes, the time elapsed would be exactly the same or very similar to the already measured durations being treated as reference values. However a major difference exists between the conditions at which the benchmark times have been gathered, and the ones at which the results of the new system are being collected. As mentioned the first test performed involved using only a single node of the cluster to execute RNA-Seq tasks, which shares the same amount of computational power with the tests that represent the already set up reference values, however in the case of the newly developed system the only node available does not store and load all the needed data from his local filesystem, as it did before when running iRAP, since it mounts a directory exported, by the server, for that matter. The pointed out difference corresponds to the dominant drawback of the solution implemented during this dissertation, as the performance gains that can be achieved will always be dependent on the existing internet connection, moreover some of the RNA-Sequencing processes load and store great amounts of data which, since the filesystem is not local, can represent a bottleneck on the system if a mediocre internet connection is being used. Taking that into consideration and taking into account that in the new system the scheduler is as well being executed, it is easily understandable that not only using just one node of the cluster, the performance of the system will not be better then before but that it is actually expected to be worse.

Table 5.3: Results for the executed tests using a 1 Node cluster

Tool	Start Time	Finish Time	Time Elapsed
FastQC	05:44:21 PM	05:46:12 PM	1min 51secs
	05:54:17 PM	05:56:08 PM	1min 51secs
	05:57:47 PM	06:00:00 PM	2min 13secs
Mean			**1min 58secs**
Bowtie2	10:45:26 AM	10:50:19 AM	4mins 53secs
	10:53:33 AM	10:58:28 AM	4mins 55secs
	11:18:38 AM	11:23:33 AM	4mins 55secs
Mean			**4mins 54secs**
Tophat2	04:22:25 PM	04:40:38 PM	18min 13secs
	04:47:56 PM	05:06:05 PM	18min 9secs
	05:10:55 PM	05:28:56 PM	18min 1secs
Mean			**18min 8secs**
HTSeq2	10:53:53 PM	11:00:06 PM	6mins 13secs
	11:07:19 PM	11:13:28 PM	6mins 9secs
	11:15:54 PM	11:22:03 PM	6mins 9secs
Mean			**6mins 10secs**
Cufflinks2	02:28:15 PM	02:51:46 PM	23mins 31secs
	02:56:35 PM	03:20:01 PM	23mins 26secs
	03:28:43 PM	03:52:02 PM	23mins 19secs
Mean			**23mins 25secs**
Cuffdiff2	11:21:38 AM	11:37:13 AM	15mins 35secs
Mean			**15mins 35secs**

In the table 5.3 shown above the results collected from the test previously mentioned, that used only 1 node of the cluster to perform RNA-Sequencing computation, are shown. From the analysis of the mentioned outcomes some important conclusions can be derived. For starters when comparing it with the results that set the benchmark, it is noticeable that for every tool examined the execution time actually increased which meets what was expected for this test case. This not only reveal that the results being obtained so far are coherent but also that the choice, previously made, of using the results from the scripts developed as reference instead of the iRAP based ones was the correct choice to make, otherwise, as explained, a direct comparison of the outcomes would not be appropriate. Moreover, in the previous stage of the system's validation process, if the benchmark values would have been set up based on the ones obtained from the iRAP direct execution, the results gathered now with only a single node, that should be worse when in comparison to the reference ones, would be better making the possible conclusions that could be taken from their comparison not relevant and in a sense inconclusive.

5.3.2 1 Server, 2 Nodes

As second test regarding the evaluation of the improvements achieved with the implementation of the chosen solution during this thesis, the execution times were measured using at this second step the two available nodes. Unlike what happened with the previous test, in the current one, the benefits, if proven to exist, of the new system can now be measured and witnessed.

Despite the drawback associated with the solution implemented, which have been explained in the previous sub-chapter, since the system takes now full advantage of the task splicing, distributed computation and the scheduling algorithm it is expected that some considerable improvements are achieved.

Table 5.4: Results for the executed tests using a 2 Node cluster

Tool	Start Time	Finish Time	Time Elapsed
FastQC	11:27:44 PM	11:28:56 PM	1min 12secs
	11:31:14 PM	11:32:28 PM	1min 14secs
	11:34:15 PM	11:36:01 PM	1min 46secs
Mean			**1min 24secs**
Bowtie2	07:55:22 PM	07:58:15 PM	2mins 53secs
	08:00:19 PM	10:58:28 AM	4mins 55secs
	08:05:17 PM	08:08:19 PM	3mins 2secs
Mean			**3mins 2secs**
Tophat2	11:15:16 AM	11:24:25 AM	9min 9secs
	11:28:21 AM	11:37:29 AM	9min 8secs
	11:40:22 AM	11:49:29 AM	9min 7secs
Mean			**9min 8secs**
HTSeq2	10:01:43 PM	10:04:39 PM	2mins 56secs
	10:19:36 PM	10:22:35 PM	2mins 59secs
	10:27:58 PM	10:30:56 PM	2mins 58secs
Mean			**2mins 58secs**
Cufflinks2	03:46:02 PM	03:53:09 PM	7mins 7secs
	04:01:54 PM	04:08:58 PM	7mins 4secs
	04:15:16 PM	04:22:25 PM	7mins 9secs
Mean			**7mins 6secs**
Cuffdiff2	11:50:09 AM	12:05:38 PM	15mins 29secs
Mean			**15mins 29secs**

In the previous table 5.4 all the results gathered on this second stage of the implemented system testing process are registered. The main purpose of performing this types of analysis to the execution times of each tool is to compare them to the reference value, that have been collected at the beginning of the validation process, so that any considerable variation can be perceived. In the table 5.5 a simple comparison, focusing only on the mean times collected both for the current test and the benchmark one is presented in order for the improvements obtained to be easily evaluated.

Aiming at easing the process of analyzing the time differences obtained, a column (speedup) have been added to the table which will represent how faster or how slower (positive or negative percentage respectively) the execution of each task became when being executed in the newly developed system.

Table 5.5: Comparison between the results obtained for the 2 node cluster and reference values

Tool	**Benchmark**	**2 Nodes**	**Speedup**
FastQC	1min 32secs	1min 24 secs	8,70%
Bowtie2	3mins 54secs	3mins 2 secs	22,20%
Tophat2	16mins 42secs	9mins 8 secs	45,30%
HTSeq2	5mins 59secs	2mins 58 secs	49,60%
Cufflinks2	23mins 1sec	7mins 6 secs	69,50%
Cuffdiff2	15mins 13secs	15mins 29 secs	Irrelevant

For starters and as expected, through the observation of the data available on the presented table, no negative percentages can be perceived under the speedup column which indicates that, after the implementation of the whole solution, no tool from the tested ones actually behave worse, as far as performance and duration are concerned. Despite the fact that the system relies on a network file system to store all the needed genome reads and files for the RNA-Sequencing processes to execute, which as previously mentioned is the most noticeable drawback of the developed system, for all the tested tools positive speedup percentages have been obtained indicating that the execution time of every one of the tools tested have decreased.

As stated *Cuffdiff* is not going to be considered for the analysis of the gains obtained with the solution implemented due to the fact that, executing it by itself, would not result in any gain or any loss of performance. Since this tool represents only one task for all the reads available, and not one task per read it was not possible to splice it into smaller sub-tasks so that a single execution of *Cuffdiff*, by itself, could take advantage of the new system's benefits.

Nonetheless focusing on all the other tools tested, as mentioned, all show positive results in the form of *makespan* reduction. However if carefully analyzed, the speedups obtained despite positive, exhibit great fluctuation. For starters the speedup obtained for the *Cufflinks* tool is far too exceptional in comparison to what could be expected using only two nodes of the cluster, which can only be explained by an error while performing the quantification of the execution times, being so, for this matter the results gathered from the *Cufflinks* tool are going to be discarded. Targeting the attention to the other tools, both *HTSeq* and *Tophat* were the ones that achieved the closer performance gains to what was expected (around 50%) when executing the solution implemented on a server and two computers executing RNA-Seq tasks. Nevertheless the smaller but even though considerable speedups obtained by *Bowtie* and *FastQC* tools does not imply that the developed solution don't work correctly with the mentioned tools, however being smaller tasks, as far as execution time is concerned, with undoubtedly less computation being done, and

which performs considerable amounts of load and store operations on the network file system, the possible gains that can be achieved one these tools are unquestionably more limited.

5.3.3 1 Server, 3 Nodes

During this chapter the cluster have been described as a set of three computers, a single server and two nodes that were going to execute all the necessary tasks related to RNA-Seq analysis. The described system, even with only two nodes to execute all workload, have been able to achieve good results as far as execution time reduction is concerned. The developed solution have been as well described as having a main drawback, which happens to be the network file system, which despite being needed for the whole system to fully work, can create some delay if a considerable amount of load and store instructions are being executed. As as a way of testing and demonstrating that this can actually be a problem creating a bottleneck under not so bright internet connections, a third test have been designed.

For the current test to be successfully performed, allowing some indisputable and self-evident conclusions to be taken, another node needed to be added to the cluster and the overall quality of the internet connections would need to be not so good as they were in the past test, where two nodes were used to perform RNA-Sequencing processes, to prove that even though the cluster have more computational power, and less tasks being executed at each node the execution time would actually be substantially higher, indicating exactly that the internet connection constituted a bottleneck in the system. Aiming at setting up the new cluster's configuration so that the described test could be performed another node have been fully configured and added to the cluster, but this time instead of plugged using an ethernet cable it was connected to the internet through a wireless connection which right away dramatically decreases the download and upload speeds of the internet connection for that node.

Table 5.6: Results for the executed tests using a 3 Node cluster

Tool	Start Time	Finish Time	Time Elapsed
FastQC	12:14:37 PM	12:19:12 PM	4mins 35secs
	12:41:40 PM	12:45:16 PM	3mins 36secs
	12:53:21 PM	12:55:58 PM	2mins 37secs
Mean			**3mins 36secs**
Tophat2	02:32:04 PM	02:45:28 PM	13mins 24secs
	02:48:53 PM	03:01:36 PM	12mins 43secs
	03:03:07 PM	03:18:16 PM	15mins 9secs
Mean			**13mins 45secs**
HTSeq2	01:39:19 PM	01:41:35 PM	2mins 16secs
	01:42:52 PM	01:45:43 PM	2mins 51secs
	01:47:51 PM	01:50:25 PM	2mins 34secs
Mean			**2mins 34secs**

In the table 5.6 just presented, as the current test does not represent the spotlight of this validation process, since it is not the one that allow the evaluation of the performance gains achieved through the reduction of the execution time, not every tool that were used in the last few tests are as well considered here. Despite the fact that the mentioned tools, which are not being brought into account for this test, have been tested for this cluster configuration, their results reveal not impressive from the point of view of the problem being addressed on this test, which is the negative impact that the quality of the internet connection can create in the developed system.

In contrast to what have previously been done in this validation chapter, for the scope of this test the results obtained are not going to be compared directly to the benchmark ones, but with the ones obtained in the previous validation stage, where two nodes where used for the execution of the RNA-Seq processes. This is going to be done since, as stated, the main purpose is to evaluate the impact, in the execution times, of a more capable cluster, computationally speaking, under a scenario where the internet connection is worse.

Table 5.7: Comparison between the results obtained for the 2 node cluster and the 3 node one

Tool	**2 Nodes**	**3 Nodes**	**Speedup**
FastQC	1min 24 secs	3mins 36secs	-61,10%
Tophat2	9mins 8 secs	13mins 45secs	-33,60%
HTSeq2	2mins 58 secs	2mins 34secs	13,50%

The last table 5.7 have the purpose of comparing the mean time elapsed, between the second phase of tests performed at the validation stage and the current one. As explained before, a negative speedup represents a higher execution time which is rendered in a loss of performance, in comparison to the system configuration that used only two nodes, while a positive value indicates the achievement of an improvement in the system's performance.

By looking at the presented table it is easily understandable that some of the speedups obtained are negative and by contrast the one regarding *HTSeq* tool exhibits a positive speedup. Even with a weaker internet connection the *HTSeq* tool is able to perform better than with a cluster constituted of two nodes, this way with the current cluster configuration the tool being mentioned took 13.5% less time to execute than it did in the previous scenario. However regarding the other two tools tested, the *FastQC* and the *Tophat* ones, the achieved speedups assumed a negative value meaning that both of them took more time to execute than before. Moreover not only the speedup for both of them is negative, but also it represents a massive negative percentage, taking up to 61%, for the FastQC tool, more time to execute than before.

Taking into consideration that from the last phase of validation, where two nodes of the cluster were used to compute RNA-Seq tasks, to the current phase, a single node fully configured was added to the cluster connecting it to the internet through a wireless connection, aiming at providing this node with a worse connection when compared to the other two that were already connected through the use of an ethernet cable, experiencing better download and upload speeds.

This way the only limiting element introduced was the internet connection in the recently added node. Having in mind that for two of the tested tools the *makespan* increased, even having the cluster more computational power available, it means that the limiting factor introduced in the system, the wireless connection, created a bottleneck which led to the increase of the execution times for both the *FastQC* and the *Tophat* tools. The fact that even under the same scenario the *HTSeq* tool was able to perform better than before, only testifies that some tools are less dependent on the internet connection due to a different ratio of the computation and the load and store operations needed, being less affected by the system's constraint currently being analyzed.

5.4 Summary

The current chapter had the purpose of validating the solution chosen and implemented, as a mean of solving or reducing the lack of performance associated with the RNA-Sequencing processes. It was crucial at an initial stage of this process to set up the environment at which the tests were going to be executed. With that in mind a cluster constituted of three computers have been set up in a way that one of them plays the role of server, which aims at executing the control module that is responsible for executing the scheduler and based on its output execute the tasks at the right node, at the right time and respecting the right execution order; While the other two nodes of the cluster have been configured to fully be able to execute the RNA-Sequencing processes and tasks.

After the configuration of the whole cluster it was of major importance to collect benchmark values, regarding the system's behaviour as it was before the implementation of the solution specified in this dissertation. This reference data collected is later being used to compare with the results obtained by the newly developed system. In order for a direct comparison to be feasible the benchmark results have been collected based on the scripts developed at the time of isolating each of the pipeline's stages so that they could be later scheduled.

The first test performed regarding the implemented solution have been done using only a single machine of the cluster to execute RNA-Seq tasks, despite the fact that the new system was designed to get the most out of distributed computation and not a single computer, this test revealed that the values being obtained were coherent indicating that the system was properly working and the benchmark values have been correctly gathered.

The main purpose of the validation stage was to evaluate whether or not the solution developed is able to outperform, diminishing the execution times, the already existing system. Focusing on the main goal of this phase, the second test designed aimed at measuring the *makespan* for some of the RNA-Seq tools used for each stage of the iRAP pipeline when being executed by the developed system using the two nodes of the cluster. From the results of this tests, it was perceivable that for every tool the execution time was always reduced, when in comparison to the collected benchmark values, and in some cases quite considerably reduced. The amount of gains, in the form of *makespan* reduction, achieved varies for each tool, being the ones with the most amount computation being done, the ones achieving the biggest gains. In some cases the speedups

obtained reached almost a 50% time reduction, being so with this test it became pretty obvious that in fact, the chosen solution was able to reduce the performance issues associated with the RNA-Sequencing processes that inevitably affect iRAP pipeline. However despite the fact that during the validation process good results, as far as performance improvements are concerned, were obtained, if more time was available to dedicate to this stage, it would have been interesting to test the speedups that could be obtained executing multiple RNA-Sequencing analysis or pipelines.

As final step of the validation stage, a test have been developed in order to demonstrate the most noticeable drawback associated with the solution implemented. Since the computation is being distributed among a cluster, the nodes executing the RNA-Seq processes are loading and storing all the data from and into a network file system that is being exported by the server, which may, under not so suitable internet connections cause performance issues. In the mentioned test an extra external node have been configured and added to the cluster and connected via a wireless which is the limiting element, since more computational power have been added to the cluster. Under the mentioned circumstances, from the three tools tested, two of them performed worse compared to the test performed using only two nodes which demonstrates that the internet connection can really become a bottleneck on the solution implemented, as the executed tools took up to 61% more time to execute with an extra computer composing the cluster, although with a slower and weaker internet connection. The fact that a tool, the *HTSeq* performed considerably better taking 13.5% less time to execute than in the previous test, indicates that the ratio of computation being performed and the load and store operations being done under a tool plays an important role on the tool's response to a not so suitable internet connection.

Chapter 6

Conclusions and Future Work

Due to the fact that medicine is an area in constant evolution and under regular investigation, the idea of starting to provide individualized medical care to the patients came to light, however for these to be possible the personal characteristics of every long-suffering needed to be known, or in other words, every person's unique variation of the human genome needed to be discovered, so that the best treatment for a specific patient could be selected. Nevertheless, the process of obtaining the individual's variation of genome is a very complex and time consuming biological process and that's the reason why recently some tools have been being designed and developed to help executing this process.

Two alternatives exist with the aim of studying gene expression as well as detecting differential expression genes, or in other words, to detect genes that are only expressed in certain circumstances. The first one to be developed was the Microarrays technique, however due to a flaw present in the mentioned methodology some background noise can affect the analysis providing not so good results as desired. Lately RNA-Sequencing have been developed getting the most out of recent biologically complex technologies, that ensures that this technique accomplishes better results, as far as gene expression studies are concerned.

Even though the RNA-Sequencing methodology is able to achieve more reliable results, when in comparison to Microarrays, this one is still limited in its use, when an analysis like this to be performed, due to the much higher costs associated with the RNA-Sequencing technique.

The main purpose of this thesis project was to improve RNA-Sequencing technique performance, in order to reduce its execution times and at the same time its execution costs, through the improvement of the iRAP pipeline. The mentioned pipeline was developed, in order to ease the process of executing and RNA-Seq analysis since several sequential stages are comprised in a single analysis, and there was the need to configure and keep track of multiple tools.

At an initial stage of this dissertation, the iRAP pipeline, as focus of this thesis, have been profiled, measuring execution times. The main goal of this profiling executed was to find the most

time consuming stages and tools being executed at each stage. From the first set of tests executed to the pipeline, *Cufflinks* as a quantification tool and *Cuffdiff* as a tool for executing differential expression analysis, were immediately identified as possible bottlenecks to the pipeline at which the dissertation could focus its intervention trying to improve their performance, reducing the *makespan* by parallelizing its execution.

Despite looking like a promising solution to the problem that needed to be solved in this dissertation, its feasibility needed to be confirmed, so no efforts would be put into a possible solution that could reveal itself impossible or not suitable. In order to analyze their viability, the tools pointed out as the most time consuming ones were invoked and their execution scrutinized while registering the CPU usage, which later revealed if the tools being analyzed were already parallelized or not. From this analysis it became pretty clear that both the tools identified as possible intervention points were already parallelized and being so the solution that was going to be applied to the problem that was being solved in the thesis project, would not return, in any case, the desired benefits.

The second approach, that represented one of the options that could be taken in order to reach the desired goals of this dissertation, consisted on the implementation of a system which core would be a scheduler aiming at providing, ideally, the optimum order, node and time at which each of the tasks, that make part on the pipeline execution, should be invoked. The use of a scheduler would only represent a valid approach to solve the problem if a cluster of computers could be used execute RNA-Seq tasks, taking the most out of the distributed computation. For starters, in order for this solution to be possible, the existing pipeline, iRAP, needed to be split since iRAP was encapsulating every action and tasks being invoked and executed during the pipeline execution, considering this, to the user the execution of the pipeline represented only a single task, which would invalidate the use of scheduling algorithm.

Bearing in mind the described system, it would be possible with its implementation for the user to split several stages of the pipeline for example into sub-tasks not depending of each other, which would allow the scheduler to map those sub-tasks to run in parallel, or even schedule multiple sets of tasks to be executed which would allow multiple RNA-Seq analysis being performed at the same time.

Choosing the scheduler algorithm to implement was a major decision that plays and unquestionably important role in the possible gains, as far as *makespan* reduction, that can be achieved. For starters a very computationally complex scheduler, even if able provides the best schedule possible, can represent a bottleneck in the system nullifying the achievable gains. On the other hand schedulers may not offer as much features or customization options that could be important in order for the achievable amount of gains to be, as much as possible, enlarged. Being so, PEFT was the chosen scheduling algorithm, to integrate the system developed, since it is able to provide a quality schedule, due to a lookahead feature, while being a low-complexity algorithm, outperforming all the other quadratic scheduling algorithms.

A control module have, as well, been developed aiming at aggregating the whole system, which represents the only software with which the user is going to have contact with. This module is going to be executed in the machine playing the role of server, with the purpose of checking all the connections that are needed for the system to fully work. The mentioned control module is even responsible for executing the scheduler and after gathering its output, through the established ssh connections, invoking the right tasks on the node of the cluster specified by the scheduler, at the specified time.

The whole system implemented, have been configured in a three node cluster with the main purpose of allowing the system to be validated. Two of the nodes have been configured to be able to successfully execute RNA-Seq processes while the other have played the role of server, exporting the network file system and executing the control module that controls the whole system behaviour. Multiple tests have been performed aiming at gathering the execution times for the new system, for some tools representing different stages of the RNA-Sequencing analysis. When in comparison to the reference results obtained, for all the tested tools, the execution times decreased from 9% in the *FastQC* tool up to 50% in the *HTSeq* one. The great variation that could be perceived in the obtained results, indicates that the performance gains that can be achieved are determined by the ratio between the amount of computation being done and the load and store instructions being, as well performed. Although, considerable speedups have been obtained in every tool tested, even using only two nodes for executing the tasks, which unquestionably represents that the implemented solution during this thesis project was able fulfill the main goal making RNA-Sequencing processes' execution much quicker than they were before.

6.1 Future Work

Despite the positive obtained results some improvements can be made in the future.

For starters the scheduling algorithm could be modified or even exchanged in order for the scheduler to be able to take into account, during the scheduling process, not only the available cluster nodes but also the available cores, for cases when a tool is being executed and does not require every core available, taking advantage of these cores that were going to be idle to execute another task. This would allow performance gains to be achieved, through the reduction of the execution times, even when executing RNA-Sequencing processes in a single machine, which in the developed system is impossible to achieve.

The control module could be modified to allow the input .XML files to specify which of the tasks, that are desired to be executed, should take advantage of the distributed computation, in other words, an improvement that could be done was to allow the user to choose which tasks should be executed in the cluster and which should be executed in only one computer, in the server for example. This would allow the cluster to be available to execute the most computationally intensive tasks, which take the most out of the distributed computation,

while the ones not getting so considerable speedups, when executed in the cluster, could be executed at the same time on a single node.

Despite the great effort put into trying to make the whole system as universal as possible, aiming at making it work in every operating system and with any tasks at all, not only RNA-Seq processes, in the control module some Unix methods are still being used which could be improved so that the system would be 100% universal.

References

[AB14] Hamid Arabnejad and Jorge G. Barbosa. List scheduling algorithm for heterogeneous systems by an optimistic cost table. *IEEE Transactions on Parallel and Distributed Systems*, 25(3):682–694, 2014.

[ABS14] Hamid Arabnejad, Jorge G. Barbosa, and Frédéric Suter. Fair Resource Sharing for Dynamic Scheduling of Workflows on Heterogeneous Systems. *High-Performance Computing on Complex Environments*, pages 145–167, 2014.

[APH15] Simon Anders, Paul Theodor Pyl, and Wolfgang Huber. HTSeq-A Python framework to work with high-throughput sequencing data. *Bioinformatics*, 31(2):166–169, 2015.

[CJB07] Louis-claude Canon, Emmanuel Jeannot, and Campus Scientifique Bp. Comparative Evaluation of the Robustness of DAG Scheduling Heuristics The University of Manchester CoreGRID Technical Report Number TR-0120 Comparative Evaluation of the Robustness of DAG Scheduling Heuristics. *Network*, 2007.

[DDS+13] Alexander Dobin, Carrie A. Davis, Felix Schlesinger, Jorg Drenkow, Chris Zaleski, Sonali Jha, Philippe Batut, Mark Chaisson, and Thomas R. Gingeras. STAR: Ultrafast universal RNA-seq aligner. *Bioinformatics*, 29(1):15–21, 2013.

[ESS+14] Pär G Engström, Tamara Steijger, Botond Sipos, Gregory R Grant, and André Kahles. Europe PMC Funders Group Systematic evaluation of spliced alignment programs for RNA- seq data. 10(12):1185–1191, 2014.

[FMB14] Nuno A. Fonseca, John Marioni, and Alvis Brazma. RNA-Seq gene profiling - A systematic empirical comparison. *PLoS ONE*, 9(9), 2014.

[FPMB14] N. A. Fonseca, R. Petryszak, J. Marioni, and A. Brazma. iRAP - an integrated RNA-seq Analysis Pipeline. *bioRxiv*, page 005991, 2014.

[GH11] Chris Gregg and Kim Hazelwood. Where is the data? Why you cannot debate CPU vs. GPU performance without the answer. *ISPASS 2011 - IEEE International Symposium on Performance Analysis of Systems and Software*, pages 134–144, 2011.

[GNT10] Jeremy Goecks, Anton Nekrutenko, and James Taylor. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome biology*, 11(8):R86, 2010.

[GTBK11] Angela Goncalves, Andrew Tikhonov, Alvis Brazma, and Misha Kapushesky. A pipeline for RNA-seq data processing and quality assessment. *Bioinformatics*, 27(6):867–869, 2011.

[HGNL12] Jun Hu, Huanying Ge, Matt Newman, and Kejun Liu. OSA: A fast and accurate alignment tool for RNA-Seq. *Bioinformatics*, 28(14):1933–1934, 2012.

[HZL+11] Songbo Huang, Jinbo Zhang, Ruiqiang Li, Wenqian Zhang, Zengquan He, Tak Wah Lam, Zhiyu Peng, and Siu Ming Yiu. SOAPsplice: Genome-wide ab initio detection of splice junctions from RNA-Seq data. *Frontiers in Genetics*, 2(JULY):1–12, 2011.

[KLS12] Vanessa M Kvam, Peng Liu, and Yaqing Si. A comparison of statistical methods for detecting differentially expressed genes from RNA-seq data. *Am J Bot*, 99(2):248–56, 2012.

[KPT+13] Daehwan Kim, Geo Pertea, Cole Trapnell, Harold Pimentel, Ryan Kelley, and Steven L Salzberg. TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome biology*, 14(4):R36, 2013.

[KRMG13] David G. Knowles, Maik R??der, Angelika Merkel, and Roderic Guig?? Grape RNA-Seq analysis pipeline environment. *Bioinformatics*, 29(5):614–621, 2013.

[LD09] Heng Li and Richard Durbin. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, 25(14):1754–1760, 2009.

[LD10] Heng Li and Richard Durbin. Fast and accurate long-read alignment with Burrows-Wheeler transform. *Bioinformatics*, 26(5):589–595, 2010.

[LS12] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with Bowtie 2. *Nat Methods*, 9(4):357–359, 2012.

[LTPS09] B Langmead, C Trapnell, M Pop, and S L Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome biology*, 10(3):R25, 2009.

[MZ13] X Ma and X Zhang. NURD: an implementation of a new method to estimate isoform expression from non-uniform RNA-seq data. *BMC Bioinformatics*, 14(1):220, 2013.

[RTN+04] Charles E Rogler, Tatyana Tchaikovskaya, Raquel Norel, Aldo Massimi, Christopher Plescia, Eugeny Rubashevsky, Paul Siebert, and Leslie E Rogler. RNA expression microarrays (REMs), a high-throughput method to measure differences in gene expression in diverse biological samples. *Nucleic acids research*, 32(15):e120, 2004.

[SD13] Charlotte Soneson and Mauro Delorenzi. A comparison of methods for differential expression analysis of RNA-seq data. *BMC bioinformatics*, 14(1):91, 2013.

[TPS09] Cole Trapnell, Lior Pachter, and Steven L. Salzberg. TopHat: Discovering splice junctions with RNA-Seq. *Bioinformatics*, 25(9):1105–1111, 2009.

[TWP+11] Cole Trapnell, Brian a Williams, Geo Pertea, Ali Mortazavi, Gordon Kwan, Marijke J van Baren, Steven L Salzberg, Barbara J Wold, and Lior Pachter. Transcript assembly and abundance estimation from RNA-Seq reveals thousands of new transcripts and switching among isoforms. *Nature Biotechnology*, 28(5):511–515, 2011.

[WGS09] Zhong Wang, Mark Gerstein, and Michael Snyder. RNA-Seq: a revolutionary tool for transcriptomics. *Nature reviews. Genetics*, 10(1):57–63, 2009.

[WMM+11] Ying Wang, Gaurang Mehta, Rajiv Mayani, Jingxi Lu, Tade Souaiaia, Yangho Chen, Andrew Clark, Hee Jae Yoon, Lin Wan, Oleg V. Evgrafov, James A. Knowles, Ewa Deelman, and Ting Chen. RseqFlow: Workflows for RNA-Seq data analysis. *Bioinformatics*, 27(18):2598–2600, 2011.

[WN10] Thomas D. Wu and Serban Nacu. Fast and SNP-tolerant detection of complex variants and splicing in short reads. *Bioinformatics*, 26(7):873–881, 2010.

[ZFLB+14] Shanrong Zhao, Wai Ping Fung-Leung, Anton Bittner, Karen Ngo, and Xuejun Liu. Comparison of RNA-Seq and microarray in transcriptome profiling of activated T cells. *PLoS ONE*, 9(1), 2014.